A STUDY OF THE P-3C OMEGA NAVIGATION SYSTEM

James Joseph Frydrychowicz

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Study of The P-3C Omega Navigation System

by

James Joseph Frydrychowicz

June 1977

Thesis Advisor: Uno R. Kodres

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A Study of The P-3C Omega Navigation System	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; June 1977
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James Joseph Frydrychowicz	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1977
	13. NUMBER OF PAGES: 92
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Naval Postgraduate School, Monterey, California 93940	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Omega
Navigation System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis is a study of the Omega navigation system as currently implemented in the P-3C aircraft. The possibility of using a microcomputer to solve the internal processing functions is investigated. Data flow graphs were applied to the velocity and navigation processing function in the Omega system. These graphs assisted in the development of the PL/M

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

code which implements the function. The four PL/M subroutines that were written can compute the velocity and navigation equations in sufficient time and with sufficient accuracy to encourage additional research into a microcomputer implementation of the remaining internal functions of the Omega system.

Approved for public release; distribution unlimited

A Study of the P-3C Omega Navigation System

by

James Joseph Frydrychowicz
Captain, United States Marine Corps
B.S., Northern Illinois University, 1967

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1977

ABSTRACT

This thesis is a study of the Omega navigation system as currently implemented in the P-3C aircraft. The possibility of using a microcomputer to solve the internal processing functions is investigated. Data flow graphs were applied to the velocity and navigation processing function in the Omega system. These graphs assisted in the development of the PL/M code which implements the function. The four PL/M subroutines that were written can compute the velocity and navigation equations in sufficient time and with sufficient accuracy to encourage additional research into a microcomputer implementation of the remaining internal functions of the Omega system.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

This thesis examines the use of a microcomputer to solve the internal processing functions of the current P-3C Update Omega navigation system. The most important reason for studying a microcomputer for the Omega navigation system is that during an excessive system load in the P-3C aircraft, the Omega signal processing is partially inhibited. Operating in this inhibited mode appeared unnecessary in a sophisticated system such as the P-3C incorporates, especially with the existing technology of microprocessors and distributed systems [Refs. 1, 2, and 3]. With a microcomputer dedicated to Omega navigation processing, this inhibited mode of operation could be eliminated and position updating information could be available continuously.

The analysis of the existing velocity and navigation processing routine utilizes the current equations in the published manual on the P-3C Update system functional description for Omega. The main tool for the analysis is the data flow graphs of the existing equations. This gives efficient and optimal PL/M code.

Section II of this thesis presents fundamental information applicable to the worldwide Omega navigation system and the AN/ARN-99(V) receiver-converter. Section III introduces the various internal processing functions of the system and presents a data flow graph depicting the parameters which apply to each of the functions. Section IV presents the analysis of the velocity and navigation processing routine with applicable data flow graphs. Section V gives the conclusion of the study.

II. THEORY OF OMEGA SYSTEM OPERATION

OMEGA is a worldwide, very low frequency (VLF), radio navigation system which utilizes the radiation from eight transmitting stations to provide global coverage to aircraft, ships, land-vehicles, and submarines for accurate and reliable positioning. Table I lists the presently available transmitting stations with appropriate letter designators, coordinates, and direction cosine values [Ref. 4]. The position of each transmitting station is shown in Figure 1.

A. SYSTEM PRINCIPLES

Each of the eight transmitting stations radiates continuous, sinusoidal wave bursts at 10.2 kHz, 13.6 kHz, and 11.3 kHz. These signals are phase locked and time synchronized to Universal Time such that all three signals start at zero value with positive slope at 0000 hours Greenwich Mean Time (GMT) and repeat at ten second intervals [Ref. 5]. Figure 2 depicts a standard ten second interval transmission pattern.

In order to understand the complexity of the Omega navigation system and the operations performed by the receiver and the computer software functions, a simplified view of the systems operation is presented first. Assume some arbitrary transmitting station is radiating on one of the three frequencies, and, at some distance d, a receiver is acquiring the signal for future processing. The

STATION	LOCATION	LAT/LONG	DIRECTION COSINE
A	Norway	66° 25' 12.39" N 13° 8' 12.65" E	SA1 = +0.91551829 SA2 = +0.39172334 SA3 = +0.09153733
B	Liberia	6° 18' 19.39" N 10° 39' 44.21" W	SB1 = +0.98221041 SB2 = +0.98117517 SB3 = -0.16588382
C	Hawaii	21° 24' 16.90" N 157° 49' 52.70" W	SC1 = +0.36283281 SC2 = -0.86296869 SC3 = -0.35162108
D	N.Dakota	46° 21' 57.20" N 98° 20' 08.77" W	SD1 = +0.72144222 SD2 = -0.10039077 SD3 = -0.68515898
E	LaReunion Island	20° 58' 26.47" S 55° 17' 24.25" E	SE1 = -0.35585299 SE2 = +0.53214875 SE3 = +0.76823587
F	Argentina	43° 03' 12.53" S 65° 11' 27.60" W	SF1 = -0.68022941 SF2 = +0.30756230 SF3 = -0.66535207
G	Trinidad	10° 42' 06.20" N 61° 38' 20.30" W	SG1 = +0.18449565 SG2 = +0.46687116 SG3 = -0.86486570
H	Japan	34° 46' 53.26" N 129° 27' 12.49" E	SH1 = +0.56547261 SH2 = -0.52409936 SH3 = +0.63683639

TABLE I. OMEGA Stations

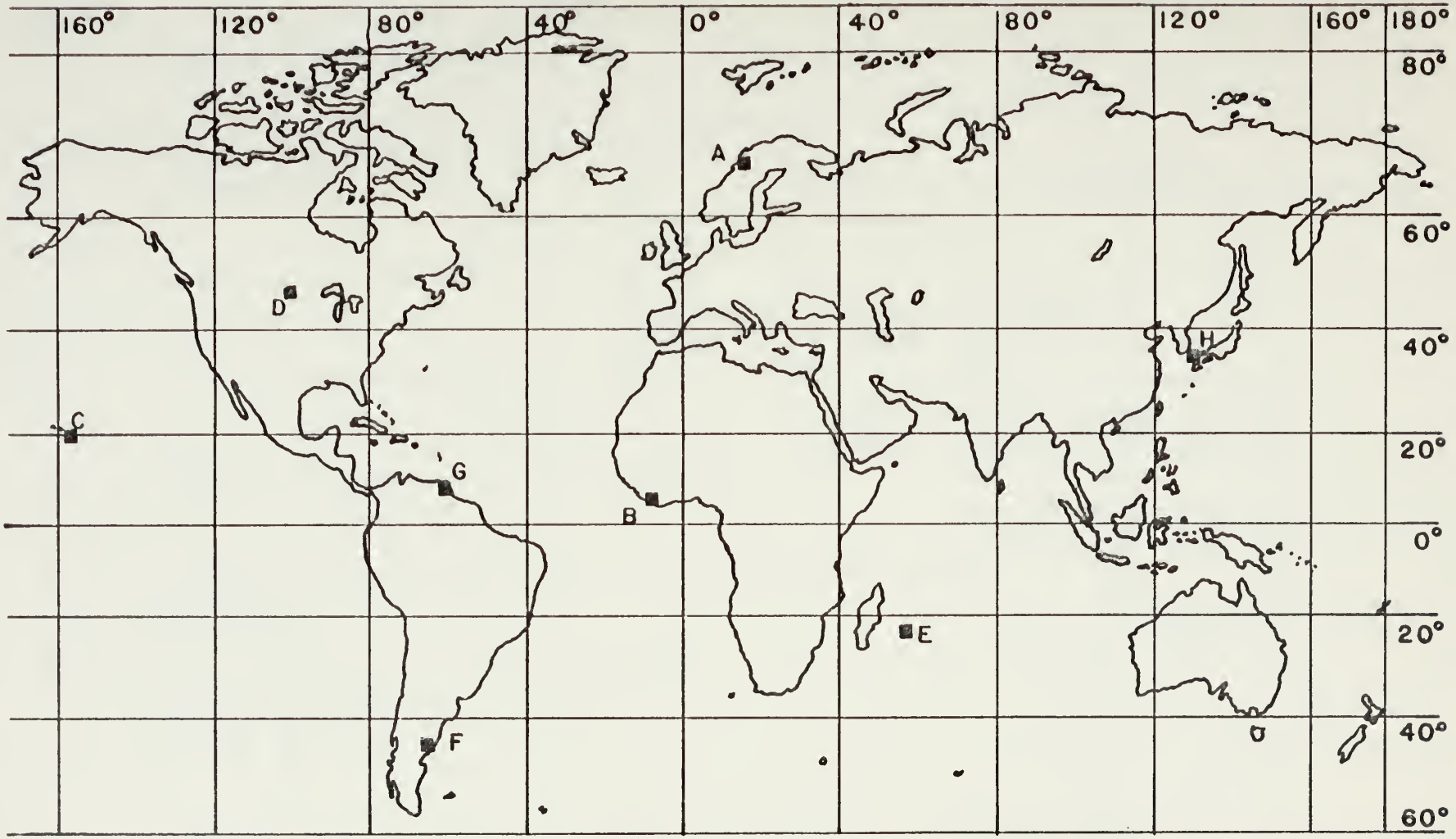


FIGURE 1 - OMEGA SYSTEM TRANSMITTER LOCATIONS

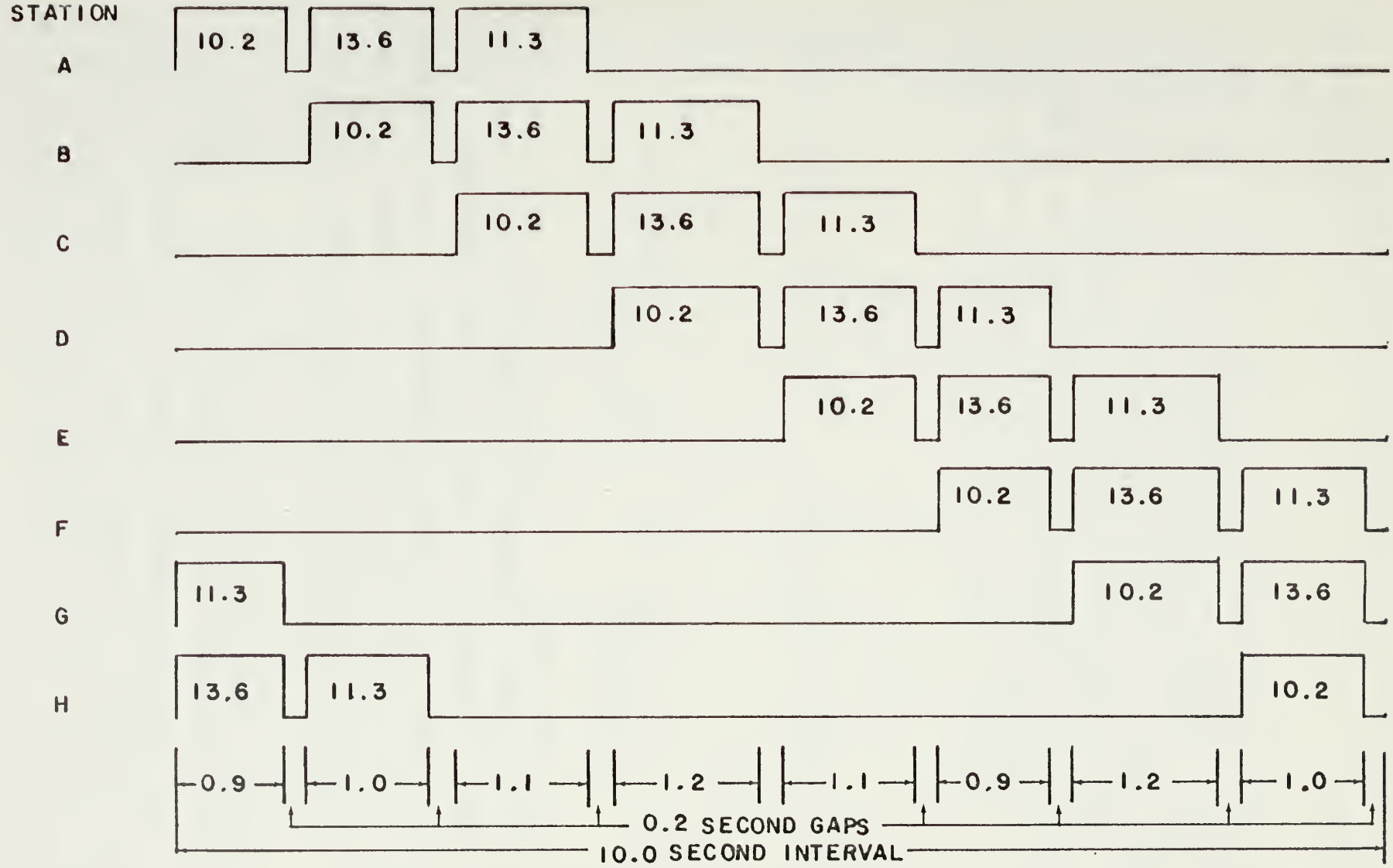
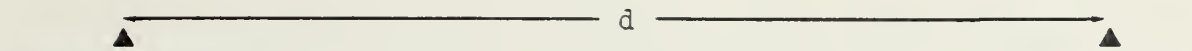
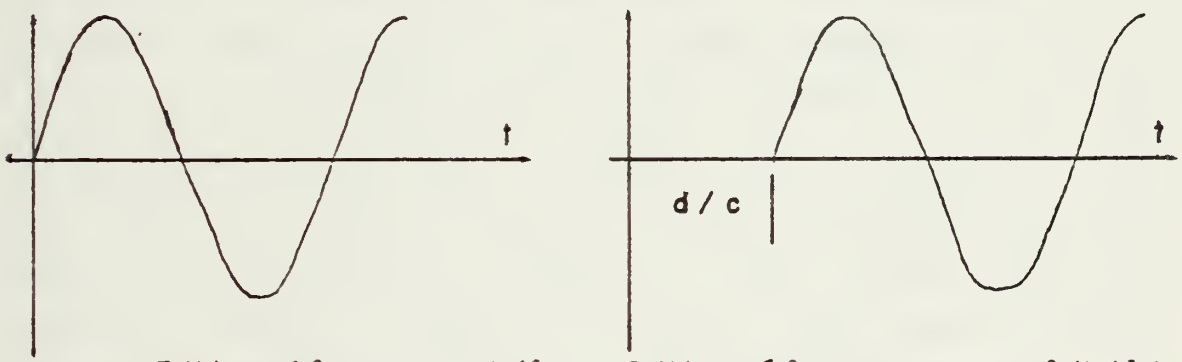


FIGURE 2 - OMEGA SYSTEM TRANSMISSION PATTERN



transmitted signal appears as a standard sinusoidal wave. Assuming no attenuation of the signal, a perfect waveguide formed by the earth and the ionosphere for the signal to propagate through, no interfering carrier or noise signal, and the transmission traveling at the speed of light, the received signal would appear as a sinusoidal wave but offset at the receiver in time t by d/c, the time necessary for the signal to travel distance d at the velocity of light c.



$$F(t) = \begin{cases} 0 & t<0 \\ \text{SIN}(\omega t) & t\geq 0 \end{cases} \qquad R(t) = \begin{cases} 0 & 0\leq t\leq d/c \\ \text{SIN}\ \omega(t-d/c) & t>d/c \end{cases}$$

However, this ideal situation is rarely achieved. The propagated signal is affected by several factors described in a subsequent section. Due to the repetitive nature of sinusoidal waves, the precise number of wave lengths must be computed between the transmitter and the receiver. Also, the absolute time t of the transmitter must be known. These variables are calculated in the combinational filter.

Position fixing with the use of OMEGA signals is accomplished by two methods. The first method utilizes phase difference information between stations and the resultant hyperbolic lines of position. The second method uses circular lines of position obtained by measuring phase with respect to a stable frequency standard. The P-3C Omega

navigation system uses the circular lines of position or rho-rho method to determine position fixes.

Of primary importance in utilizing the OMEGA VLF signals for the rho-rho method are the ability to detect and measure the phase of a received signal, and the ability to predict the phase of the signal. In order to detect and measure the phase of a received signal using the rho-rho method, the first process performed in the Omega navigation system is to relate the internal clock of the computer with the transmission burst pattern of the stations. This synchronization process is described in more detail in a subsequent section. The overall effect allows the computer program to know what station and frequency is being processed at any internal system time t. If all three frequencies from all eight stations could be received at any point on the earth's surface, the ten second reception pattern would appear as in Figure 3. For a more realistic reception pattern, Figure 4 depicts the three closest transmitter stations to Hawaii, and Figure 5 represents the signal pattern an aircraft would receive at Hawaii.

Once synchronization is completed, the Omega navigation system commences to process the received signals through various software filters in order to measure the phase. The filters which process the received signal are composed of three burst filters (ie. one for each frequency) and 24 tracking filters (ie. one for each of the three frequencies on each of the eight stations). The primary function of the burst filters is to calculate the phase measurement and the phase variance for each frequency and station received by the receiver. The tracking filter, on receiving the phase measurement and variance of the phase measurement from the burst filter, utilizes these values to acquire a more accurate estimate of phase and phase variance over several ten second time intervals.

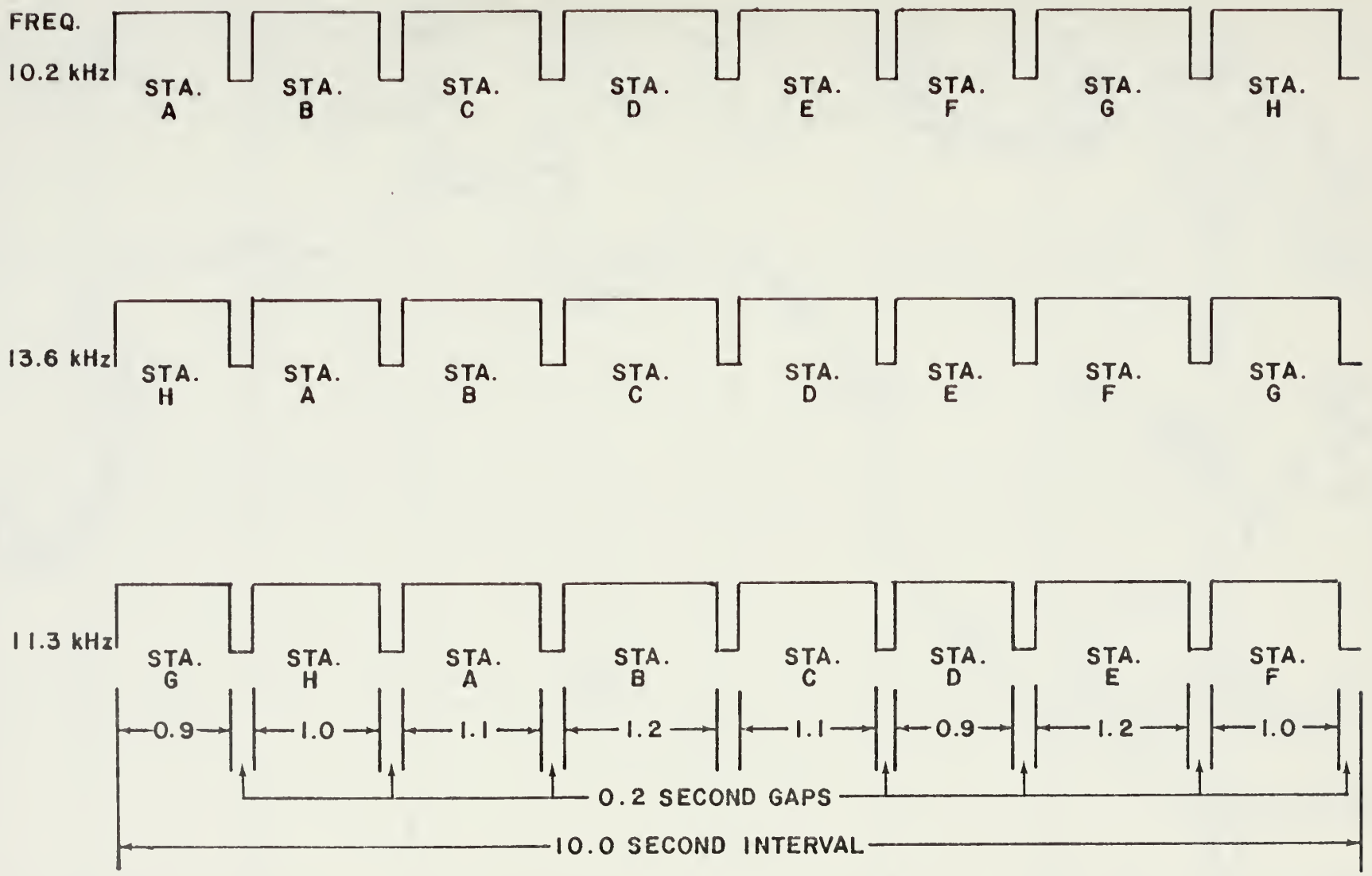


FIGURE 3 - OMEGA SYSTEM RECEIVER PATTERN

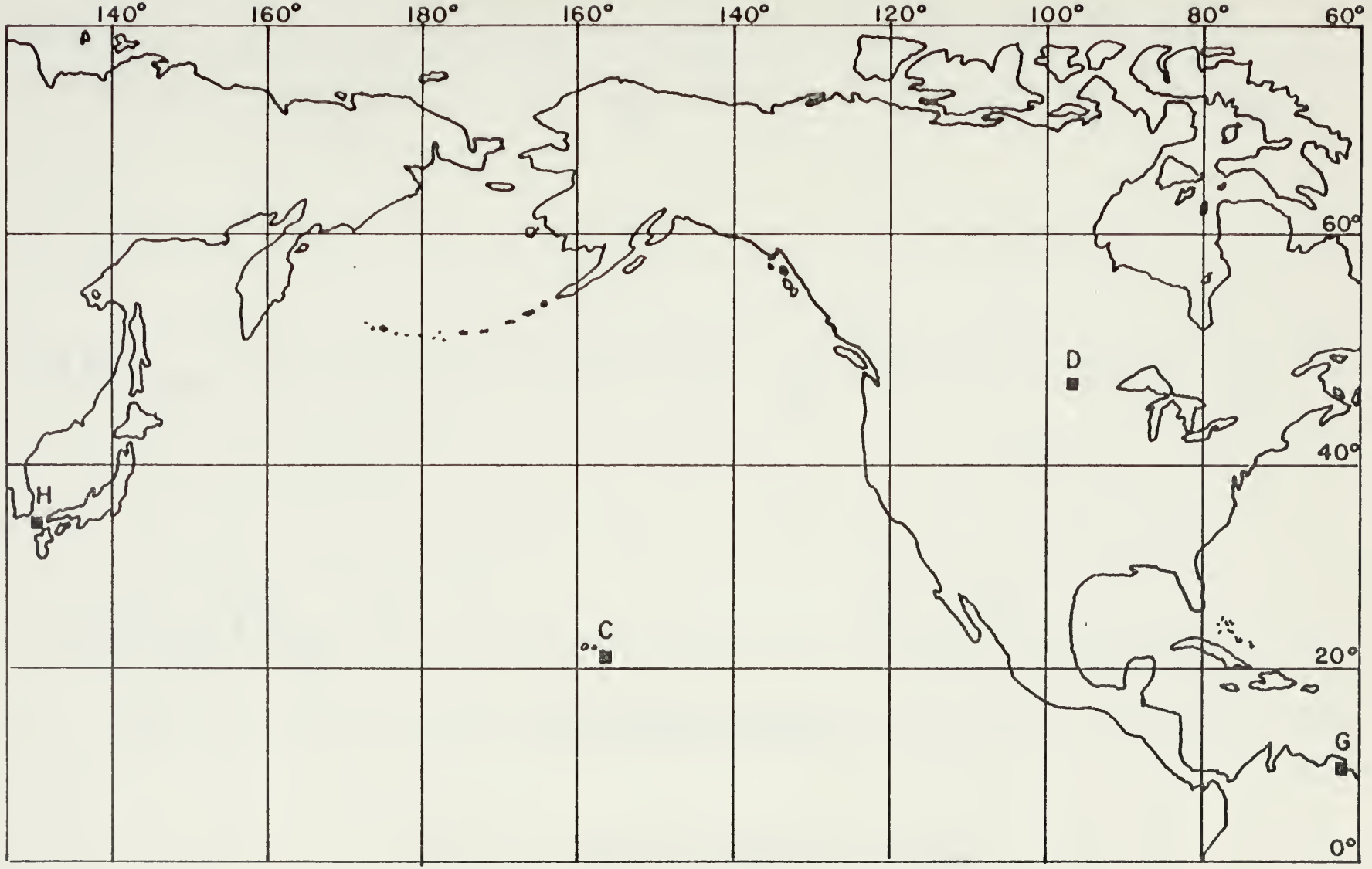


FIGURE 4 – SPECIFIC TRANSMITTER LOCATIONS

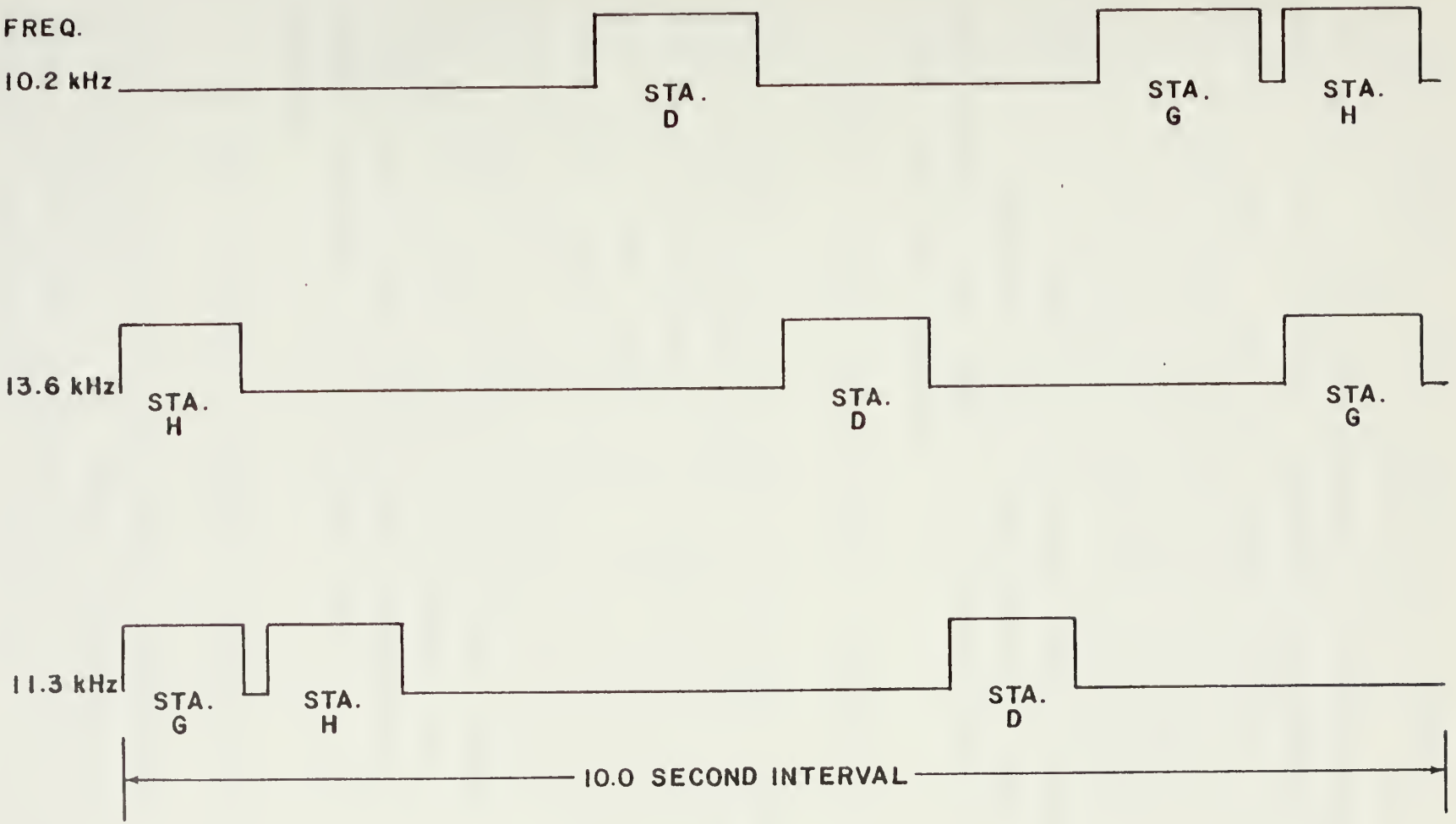


FIGURE 5 – SPECIFIC RECEPTION PATTERN

The tracking filter performs two different updates on the burst filter data. The first update is a time update. This update is required because of the displacement of the aircraft receiver between successive measurements of the received signal. The time update procedure projects the last estimate of phase through time so that it reflects the new position of the aircraft. These procedures are defined as rate aiding and are dependent on velocity sources external to the Omega system. In estimating the change in position, it is possible to estimate what the change should be for a particular station and frequency. This predicted phase change is then added to the last estimate of phase to create a new phase estimate.

$$\phi TRK(t) = \phi TRK(t-1) + [\ \Delta\phi DR(t) + \Delta\phi ERR(t-1)\ ] * \Delta t$$

where $\Delta\phi DR(t)$ is the phase rate which represents the average rate of change of phase along the arc connecting a station and the aircraft, and $\Delta\phi ERR(t-1)$ is the estimate of the error in the phase rate and is calculated in the measurement update section. The second update is a measurement update, and is performed every time the burst filter supplies a measurement of phase and phase variance to the tracking filter. This measurement update procedure combines the rate aiding phase estimate from the tracking filter with the burst filter phase measurement to produce a better estimate of the phase based on the following equation.

$$\phi = \phi JK - \phi TRK(t) + [\ \Delta\phi DR(t) + \Delta\phi ERR(t)\ ] * (0.425)$$

where ϕ is defined to be the angular difference of phase. The combination of the phase measurement from the burst filter and the phase estimate from the tracking filter is a weighted average, where the coefficients are computed using

the phase variance from the burst filter ($\sigma^2\phi JK$) and the phase variance from the tracking filter ($\sigma^2\phi TRK$).

$$\phi TRK(t) = \phi TRK(t-1) + \text{ARCTAN}\left[\frac{(\sigma^2\phi TRK * S)}{((\sigma^2\phi JK + \sigma^2\phi TRK) * C)}\right]$$

where $S = \text{SIN}\ \phi$ and $C = \text{COS}\ \phi$.

Once the measurement of phase has been calculated, it is supplied to the combinational filter for processing along with a phase estimate calibrated in the propagation prediction procedure. With the measured phase and the predicted phase available for position fixing, the last requirement to be implemented for utilization of the rho-rho technique of position fixing is to synchronize the receiver oscillator with the transmitter oscillator. This is accomplished in the combinational filter by computing the time difference between the transmitting station phase and the receiver oscillator phase.

1. Signal Coverage

The VLF signals utilized by the OMEGA system are transmitted over extremely long distances of the earth's surface, because they are propagated through a natural wave guide formed by the surface of the earth and the ionosphere. Due to the advantageous properties of high phase stability and low attenuation rates of VLF signals, no modulation of the signals is required prior to transmission. Because these signals are unmodulated, only their relative position in any ten second time interval enables receivers to identify the station being received. Of primary importance in utilizing the OMEGA VLF signals are the ability to detect and measure the phase of a received signal, and the ability to predict the phase of the signal. Several factors have an effect on

the propagation of VLF signals and consequently the ability of an airborne Omega system to receive and process the signals for accurate positioning [Ref. 6]. These factors are described in the section pertaining to propagation prediction.

2. Receiver-Converter

The AN/ARN-99 receiver was developed by the Electronics Division of the Northrop Corporation in Hawthrone, California. It was designed for operation with all eight transmitting stations and to provide a continuous update of aircraft position. The receiver-converter installed in the P-3C consists of several modular assemblies as shown in Figure 6. The antenna, composed of two loop antennas fixed at right angles to each other and mounted at 45° angles to the aircraft center line, is controlled by the burst filter function in the computer through the antenna switching matrix. The three heterodyne receivers have RF and IF sections for processing of the input signals. The requirements for filtering the received signals in these sections are very stringent. The filtering in the RF section removes the RF image, including all frequencies in the area of the harmonics of the local oscillator. The narrow band filtering in the IF section provides for the rejection of interfering carriers. In addition, limiters are used for controlling the dynamic signal levels, and a bandpass filter is used to remove the harmonics created by the previous limiters. The resultant output of the RF and IF filtering sections leaves the fundamental frequency as a sinusoid. This output is fed into a sine and cosine correlator which utilizes a square reference signal to produce an output of two direct current signals. One dc signal is proportional to the sine of the burst phase, and the other dc signal is proportional to the cosine of the

burst phase. These dc signals, which represent the measured phase shift, enter the analog to digital converter and are transmitted to the computer section through the receiver-converter/computer interface box. The precision frequency generator provides for the reference signals and test signals necessary in the measuring of the phase difference of the receiver signal [Refs. 7, 8].

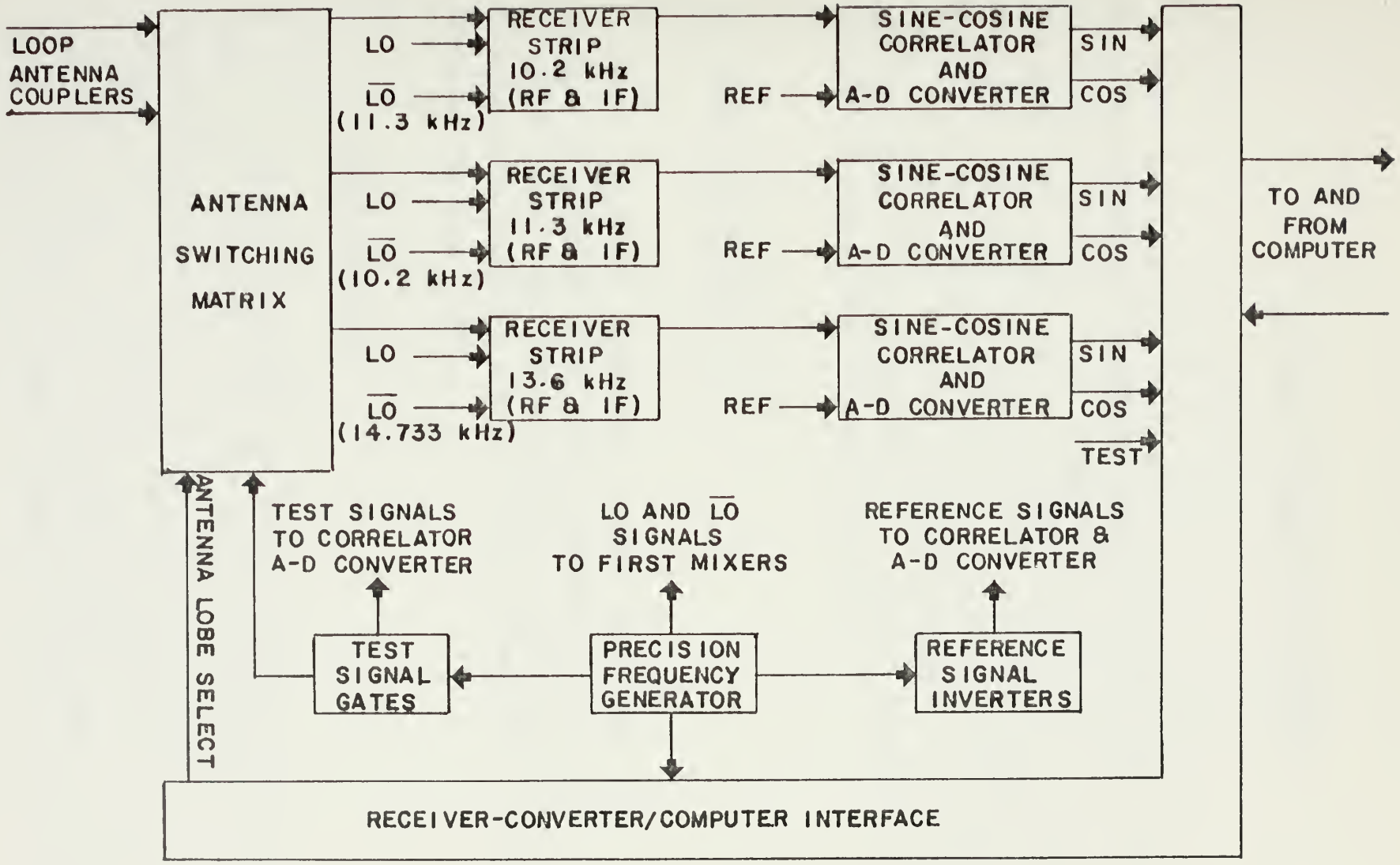


FIGURE 6 - RECEIVER-CONVERTER/COMPUTER INTERFACE

III. INTERNAL PROCESSING FUNCTIONS

The Omega navigation system in the P-3C aircraft is one of five navigation subsystems which comprise the avionics navigation system. The remaining subsystems, in their priority of use, are a primary inertial, a secondary inertial, a doppler radar, and an air data computer. The selection and utilization of each subsystem is dependent on the control of the Navigator/Communicator and on the availability of the subsystem. A block diagram of the navigation subsystems is shown in Figure 7.

A. OMEGA PROCESSING

The internal processing functions of the Omega navigation system are shown in Figure 8 [Ref. 4]. A more detailed description of the internal processing functions is presented in Figure 9 showing the pertinent data transferred between the functions.

1. Initialization

Initialization of the Omega navigation system begins with the insertion of the date and time entries by the Navigator/Communicator and subsequent activation of the initial on top condition or mark aircraft position. The date and time entries are used within the propagation prediction module to calculate an estimate of the phase value derived from current aircraft position at the moment

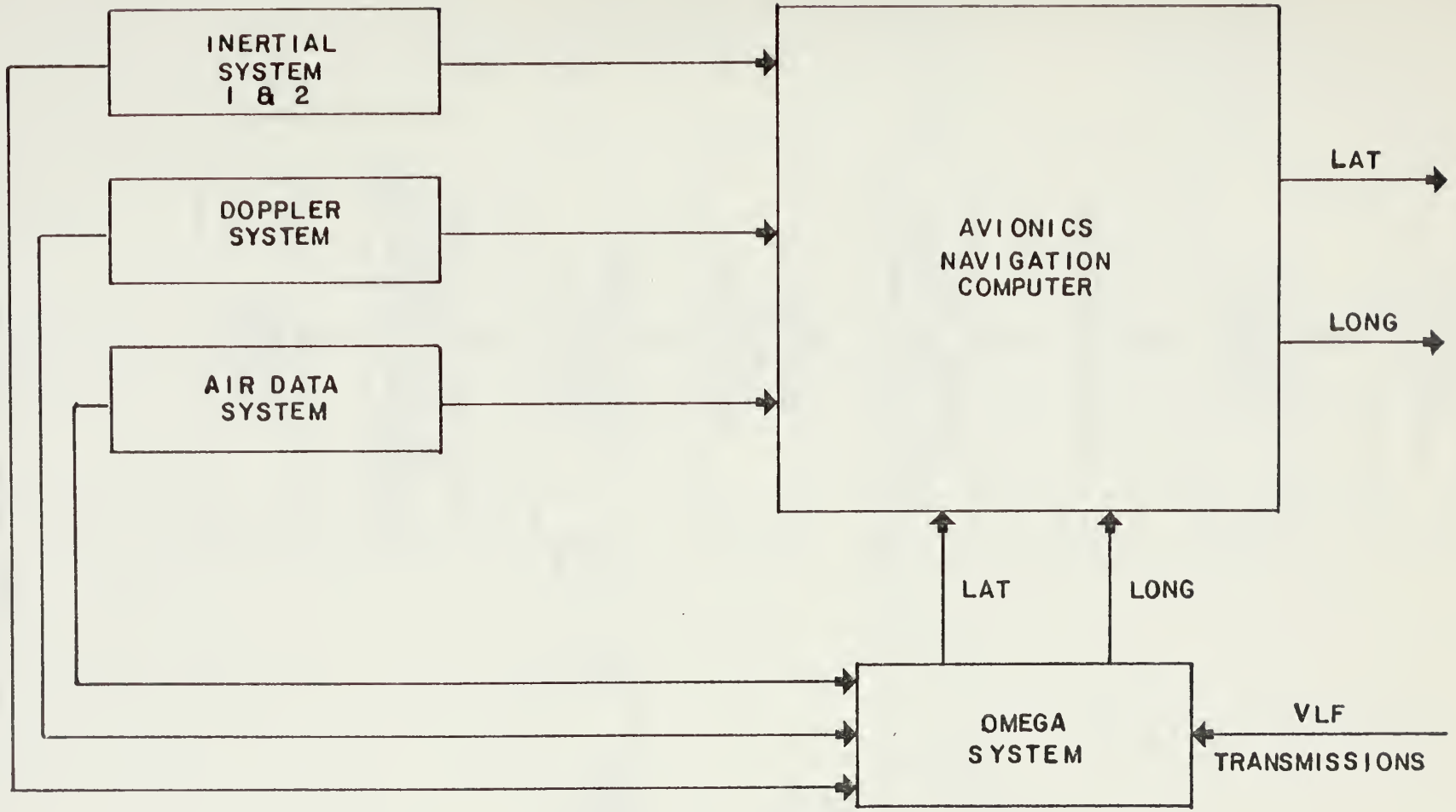


FIGURE 7 - NAVIGATION SYSTEM BLOCK DIAGRAM

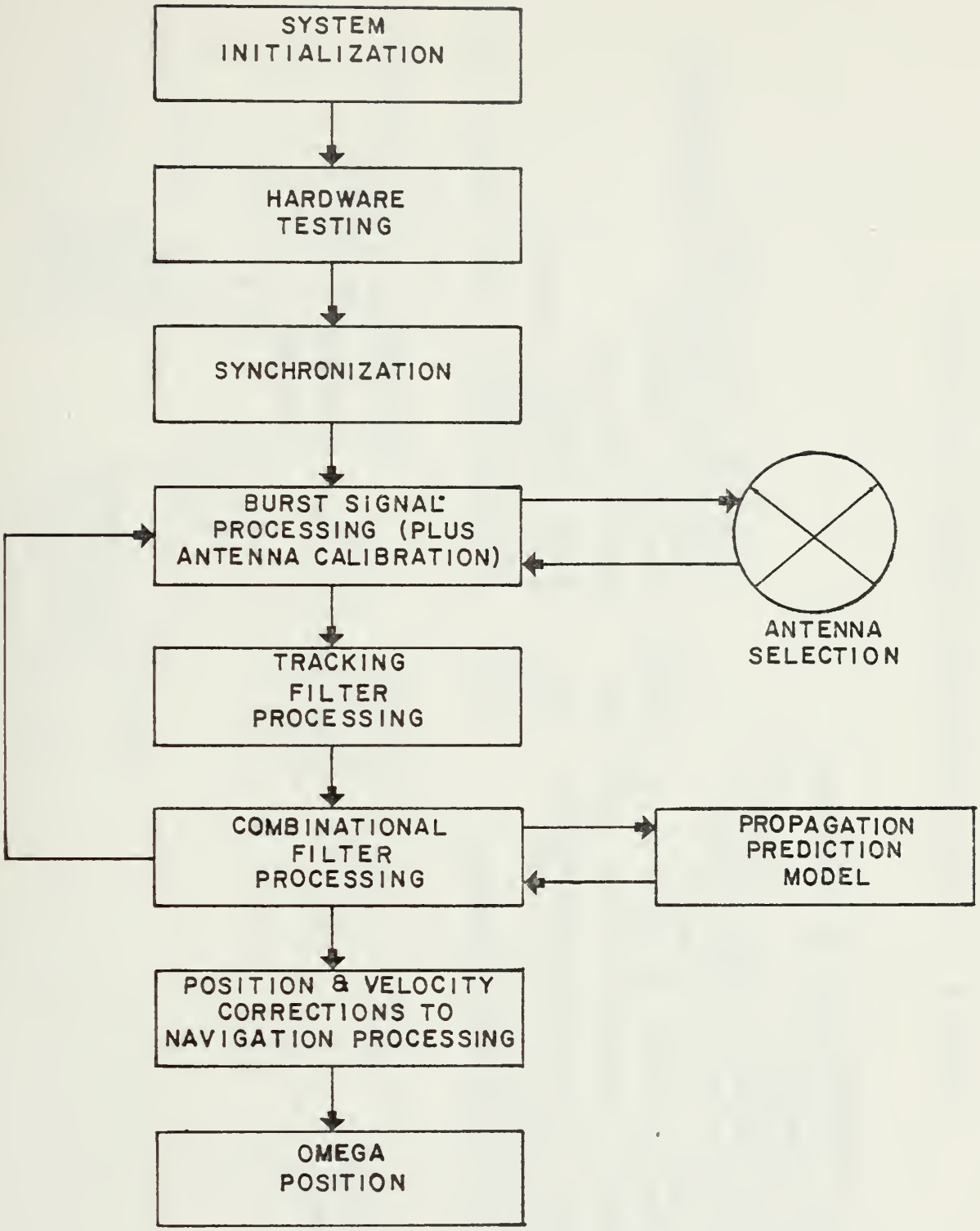
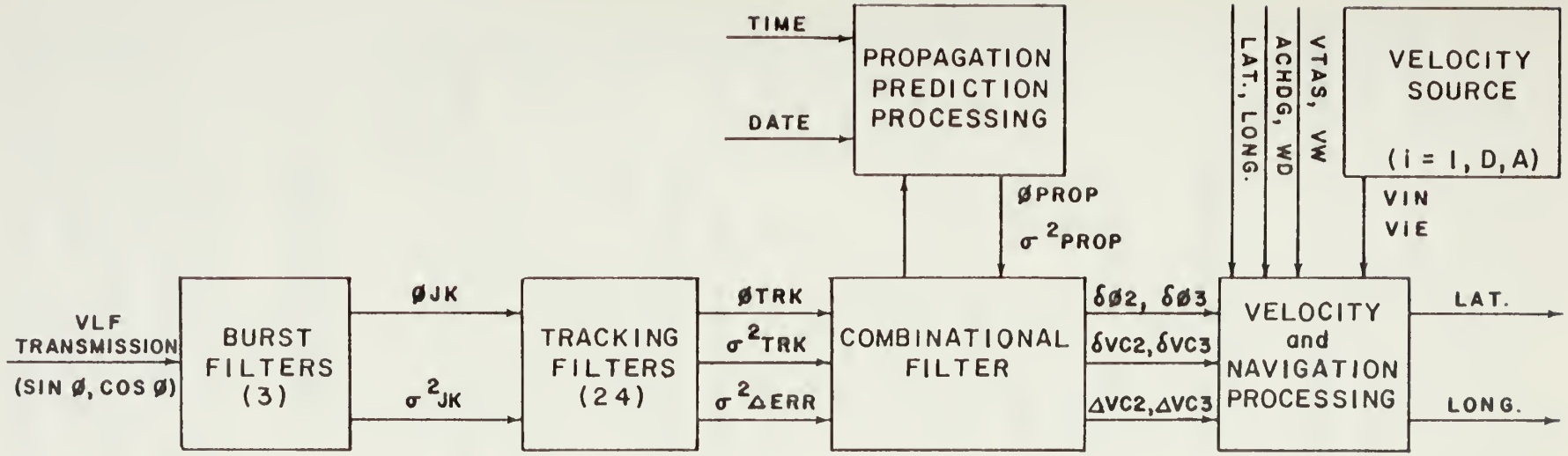


FIGURE 8 - INTERNAL OMEGA PROCESSING FUNCTIONS



ØJK - BURST FILTER PHASE MEASUREMENT OF BURST J FREQUENCY K

σ^2JK - VARIANCE OF BURST FILTER PHASE

ØTRK - TRACKING FILTER PHASE MEASUREMENT

σ^2TRK - VARIANCE OF TRACKING FILTER PHASE

σ^2 ERR - VARIANCE OF ERROR IN PHASE IN TIME

ØPROP - PROP. PRED. PHASE ESTIMATE

σ^2PROP - VARIANCE OF PROP. PRED. PHASE

δØ2, δØ3 - POSITION ERRORS ALONG THE -R2 AND -R3 DIRECTION OF THE SYSTEM POSITION MATRIX R

δVC2, δVC3 - VELOCITY ERRORS IN THE EAST AND NORTH COMPONENTS OF VELOCITY

ΔVC2, ΔVC3 - ACCUMULATION OF THE ESTIMATES OF δVC2 AND δVC3

VTAS - TRUE AIRSPEED VELOCITY

VW - WIND VELOCITY

WD - WIND DIRECTION

ACHDG - AIRCRAFT HEADING

VIN, VIE - VELOCITIES NORTH AND EAST FROM AVAILABLE SOURCE (Ie. INERTIAL, DOPPLER, OR AIR DATA)

FIGURE 9 — DATA FLOW GRAPH OF INTERNAL FUNCTIONS

of initialization. These phase values are used to initialize the tracking filters. The variance values within the combinational filter are also initialized at this time, and the position matrix R, utilized in the velocity and navigation processing function, is initialized based on current aircraft latitude and longitude.

2. Hardware Testing

Hardware testing begins immediately after the termination of the initialization function, and also after each restart. The main objective of the hardware testing module is to determine the status of the Omega receiver/converter. Additionally, hardware testing provides a method of assuring proper operation of all Omega equipment. The testing involves a series of subtests which validates the operation of the input/output interface between the receiver/converter and the computer. The subtests, which are described in greater detail in Ref. 4, are a communication subtest, a coherence status subtest, an Omega output subtest, an Omega input subtest, a phase angle to digital subtest, a phase counter subtest, and a RF/IF subtest. An oscillator drift subtest, which is classified as a hardware test, is not implemented until single frequency processing commences in the combinational filter routine.

3. Synchronization

Synchronization of the Omega navigation system to the Omega transmission pattern is the most critical function executed by the navigation system. By aligning itself with the ten second interval transmission pattern, the Omega navigation set processes the correct VLF signal bursts in

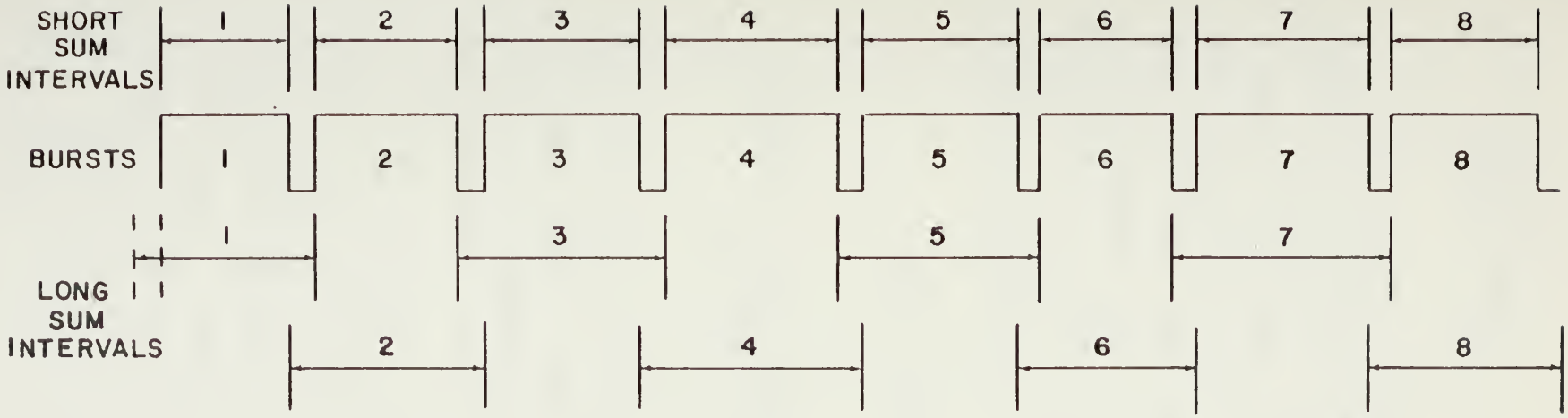
subsequent routines. Synchronization is accomplished by utilizing the variable length transmission time intervals of the various frequencies and a differential correlation technique. The first assumption made in the routine is that all eight stations that are presently in operation are operating on all three frequencies. The routine, starting at some random time, accepts from the receiver-converter a ten second interval of data composed of 100 sine and 100 cosine values over contiguous 100 millisecond intervals. One hundred correlation coefficients are calculated for the frequency under consideration utilizing the difference between short sum and long sum intervals and adding these differences over the eight count burst pattern. Figure 10 depicts the short sum and long sum intervals over a ten second interval pattern, and the limits of the short and long sums. The correlation coefficient formula is given below.

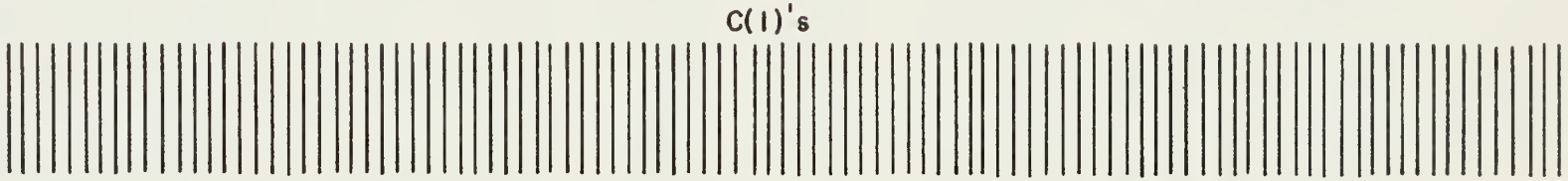
$$C(I) = \sum_{J=1}^{8} \left[\frac{\left(\sum_{K=I+JL}^{I+JU} X(K) \right)^2 + \left(\sum_{K=I+JL}^{I+JU} Y(K) \right)^2}{JU-JL+1} - \frac{\left(\sum_{K=I+JL-2}^{I+JU+2} X(K) \right)^2 + \left(\sum_{K=I+JL-2}^{I+JU+2} Y(K) \right)^2}{JU-JL+5} \right]$$

$$(I=1,100)$$

The values X(K) and Y(K) are the sine and cosine values respectively received from the receiver-converter. Each of the confidence coefficients can be considered to be a measurement of how well the received ten second burst pattern matches the transmitted burst pattern. The higher the coefficient value, the greater confidence is associated to the index number I being the starting point of the transmitted burst pattern within ± 50 milliseconds. The four largest confidence coefficients are saved in descending order of magnitude, along with their respective index number I. This index number is used in determining the starting point of the ten second burst pattern. Various conditions

relating to the four highest confidence coefficients and index numbers are tested in order to determine the success or the failure of the synchronization attempt. If the current confidence level fails, the next sequential frequency is utilized to acquire the next ten second interval of data. The synchronization routine continues in this mode until a successful synchronization occurs or until the Navigator/Communicator intervenes.





BURST J	JL	JU
1	0	8
2	11	20
3	23	33
4	36	47
5	50	60
6	63	71
7	74	85
8	88	97

FIGURE 10 – SYNCHRONIZATION PARAMETERS

4. Burst Filter

The burst filter routine commences upon successful completion of synchronization. This routine calculates the phase measurements and the variance of these phase measurements for each station and frequency received by the Omega navigation set. It also selects the optimal antenna inputs and test data for the Omega receiver, maintains synchronization with the transmitted burst pattern, and computes the phase measurement and its confidence measurement based on the receiver phase difference and test data from the receiver. The equations are:

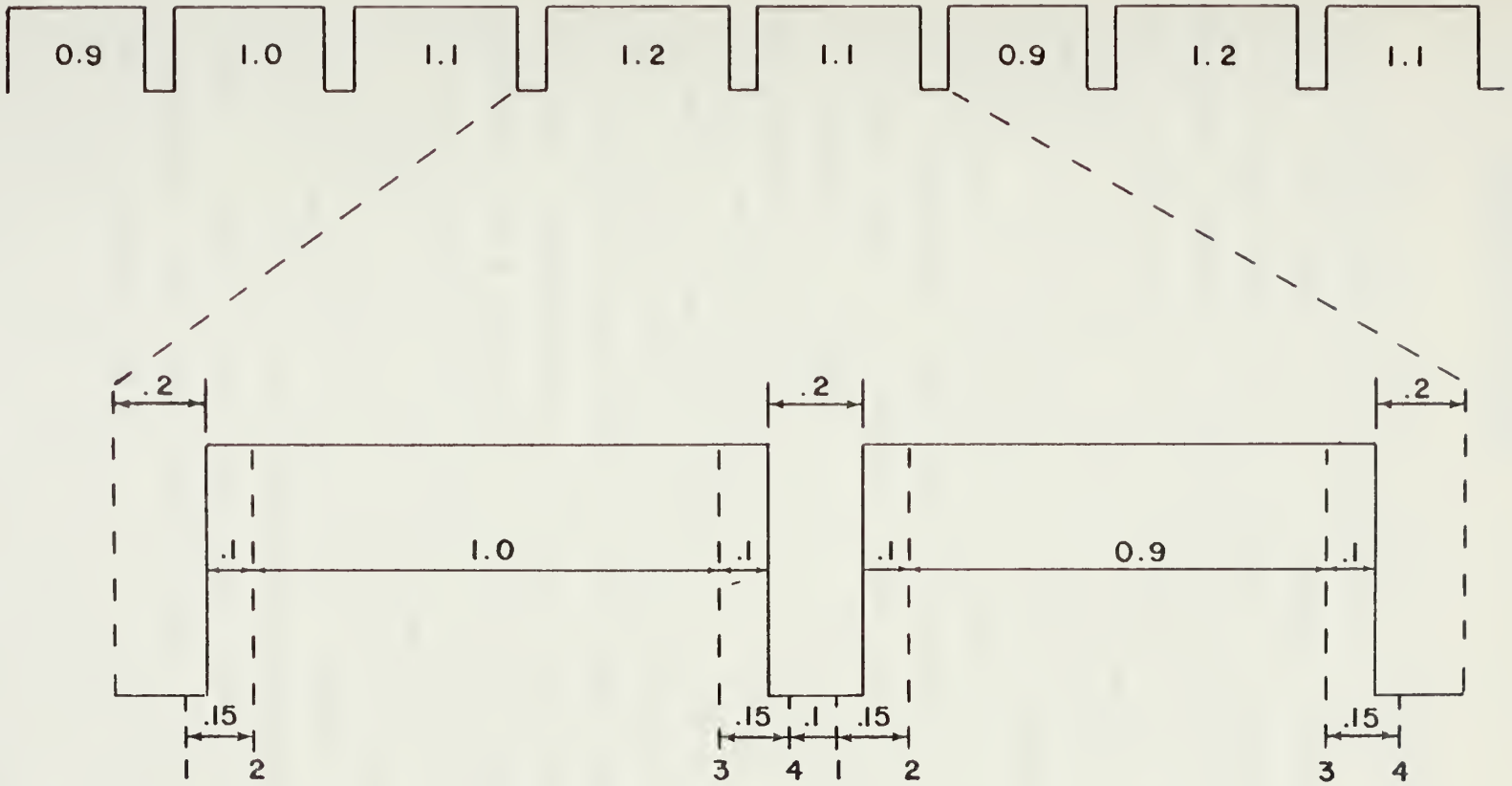
$$\phi JK = \text{ARCTAN}\left[\frac{(XbJK)}{(YbJK)}\right] - \phi OK + \phi'$$

$$\sigma^2 JK = \left[\frac{QtK * (\Delta tJ - QJK)}{2 * \Delta tJ * (QJK - QtK)}\right] + [(0.005) * (2\pi)]^2$$

where ϕ' is a phase correction (0° or 180°) added to the received signal to account for error in reception, ϕOK is the phase angle offset for frequency K computed in the calibration measurement, and XbJK and YbJK are burst measurement sums corrected for phantom error.

To accomplish these calculations, each of the eight bursts and associated slots are divided into intervals of time as shown in Figure 11. The start burst and end burst intervals are periods of time where the sine and cosine data are not used. These intervals are considered transient periods or waiting periods. During the burst interval, the sine and cosine values received are used in the calculation of the phase measurement and variance of the phase measurement. In the test interval, various measurements

are computed to improve the accuracy of the phase measurement, and for the issuing of antenna selection and test selection commands. Reference 4, pages 86 to 101, describes in detail the computations involved in calculating the phase and the phase variance for each station and frequency used by the Omega navigation system.



INTERVAL		TIME	
1 - 2	START BURST	1	ISSUE ANTENNA SELECTION COMMANDS; END NOISE/CALIBRATION MEASUREMENTS; INITIATE PROCESSING TEST DATA
2 - 3	BURST	2	START USEABLE BURST MEASUREMENTS
3 - 4	END BURST	3	ISSUE NOISE/CALIBRATION TEST COMMANDS; END USEABLE BURST MEASUREMENTS; INITIATE BURST DATA PROCESSING
4 - 1	TEST SLOT	4	START OF USEABLE TEST/NOISE/CALIBRATION MEASUREMENTS

FIGURE 11 - BURST FILTER TIME INTERVAL

5. Tracking Filters

Once the burst filter processing has acquired a phase measurement and an estimate of the variance of the phase measurement for a specific station and frequency, these values are used as inputs to only one of the 24 tracking filters. Each filter is similar in the processing of the data, but each is different in its computational constants for the various frequencies and stations. A simplified description of a tracking filter is that it is utilized to acquire a better estimate of the phase measurement by averaging the outputs of the burst filter over successive ten second intervals. The tracking filter routine, using the measured phase from the burst filter and an updated estimate of phase based upon previous measurements and dead reckoning information, combines the measured and estimated phase and utilizes a weighted average to compute a new phase estimate and phase variance estimate. Also computed within the tracking filter is the variance of the estimate of the error in the phase rate of change due to aircraft displacement in time. Once these three values are computed for a particular station and frequency, a test is made to determine the accuracy of the phase measurement. If the variance of the phase measurement is less than or equal to $(0.06\ \pi$ radians$)^2$, the phase measurement is considered accurate enough for computations within the combinational filter. After three successive measurements and successful tests, a flag is set indicating that the data is valid and available, and the combinational filter reads the outputs from the tracking filter. Once the outputs have been read, the count of successive measurements is reset to zero, the variances of the phase measurements are set to the initialization values, and the data valid and available flag is cleared. If at any time in the tracking filter the

variance of the phase measurement exceeds $(0.06\pi$ radians$)^2$, the count of the successive measurements is reset to zero, and the data valid and available flag is cleared.

6. Combinational Filter

The input values to the combinational filter from the tracking filter are the refined estimates of phase measurement, ΦTRK, phase variance, $\sigma^2\Phi$TRK, and phase rate variance, $\sigma^2\Phi$ERR. The inputs from the propagation prediction routine are the estimates of phase, ΦPROP, improved for propagation effects and phase variance, $\sigma^2\Phi$PROP. Within the combinational filter routine, these input values are statistically combined to provide the best estimates of system position and velocity. The combinational filter routine provides for the conversion from phase to geodetic coordinates (latitude and longitude), and also determines the phase and frequency differences between the receiver oscillator and the transmitted OMEGA signal. Once the Omega receiver oscillator is calibrated, the Omega navigation system operates when only two transmitting stations are accessible. In addition, the combinational filter is used for lane determination through the technique of multiple state vectors.

a. Description

The combinational filter is a Kalman filter. Reference 4 describes the Kalman filter technique to filtering and prediction as a linear, recursive, minimum variance filter. R. G. Brown and L. L. Hagerman [Ref. 9] describe a Kalman filter as simply a means of estimating the various states of a random process from a set of discrete measurements having a known linear connection to these

states. It was further noted in Ref. 10, that a Kalman filter operates only on the system errors and not on total quantities such as position and velocity. The basic concepts involved are those of state, state transition, measurement, and optimal weighting [Ref. 11]. The states of the combinational filter are differentials of system variables [ie. error in position ($\delta\phi 2$ and $\delta\phi 3$), oscillator start time (t_o), rate of change in oscillator start time ($\dot{t}_o$), and error in velocity ($\delta VC2$ and $\delta VC3$)]. The following chart describes the elements of the state vector X [6 x 1].

ELEMENT	SYMBOLS	MEANING
1	$\delta\phi 2$	Position error along the -R2 direction of the system position matrix R.
2	$\delta\phi 3$	Position error along the -R3 direction of the system position matrix R.
3	t_o	Time difference between the transmitting station phase and the receiver oscillator phase.
4	$\dot{t}_o$	Time rate of change of t_o.
5	$\delta VC2$	Error in the east component of velocity which is error along the R2 direction of the system position matrix R.
6	$\delta VC3$	Error in the north component of velocity which is error along the R3 direction of the system position matrix R.

The state of the system is described by the solution of linear vector differential equations depicting system error growth. The Kalman filter method linearly combines the previous estimate of the state vector, X, with a measurement to approach a minimum variance estimation. This minimum variance estimate is then time updated until the following measurement. The measurement calculation is explained by the following equations.

$$Y = \phi TRK - \phi PROP$$

where Y is the difference between the tracking filter phase measurement and the propagation prediction phase estimate of the phase measurement.

$$Residual = Y - M*X(t-1)$$

where Residual (RES) is the difference between the calculated error in the measurement, Y, and the system estimate of the error, M*X(t-1). The state vector X(t-1) is the previous estimate of the state vector, and M [1 x 6] is the measurement matrix described below.

ELEMENT	EQUATIONS	MEANING
1	$\dfrac{EMER * \sum_{J=1}^{3} R2J*SIJ}{K * SIN(\phi D)}$	EMER is the earths mean equatorial radius = 20925741.47 ft R2J and R3J are the elements of the R2 and R3 vectors of the position matrix R.
2	$\dfrac{EMER * \sum_{J=1}^{3} R3J*SIJ}{K * SIN(\phi D)}$	SIJ's (I=A,B,..,H) are the elements of the station direction cosine values.
3	$\dfrac{VL}{\lambda K}$	VL is the average propagation velocity, and λK (K=1,2,3) is the wave lengths of the frequencies 10.28, 11.3, 13.6kHz

where $\phi D = ARCCOS(\sum_{J=1}^{3} R1J*SIJ)$ and the remaining elements of M are zero. New, minimum variance, state vectors are produced by using the following equations.

$$Xnew = Xold + b * (RES)$$

where b [6 x 1] is the optimal weighting vector matrix.

This cptimal weighting vector, b, is established by means of time propagation of system error growth in combination with information contained within the measurement. The weighting is a function of the covariance matrix, P [6 x 6], of the difference between system error state vector, X, and the average state vector Xavg.

$$P = E * [(X - Xavg) * (X - Xavg)^T]$$

where E is the expected value of the difference between the value of the difference between the value for the real state vector X, if all conditions were known, and the average state vector, Xavg. The following chart describes the elements of the covariance matrix P.

ELEMENT	SYMBOLS	MEANING
P11	$\sigma^2 \Phi 2$	The position variance for the position error, $\delta\Phi 2$, along the -R2 direction
P22	$\sigma^2 \Phi 3$	The position variance for the position error, $\delta\Phi 3$, along the -R3 direction
P33	$\sigma^2 t_o$	The variance of oscillator phase offset
P44	$\sigma^2 \dot{t}_o$	The variance of oscillator drift rate
P55	$\sigma^2 VC2$	The variance of the error in the east component of velocity $\delta VC2$
P66	$\sigma^2 VC3$	The variance of the error in the north component of velocity $\delta VC3$

All remaining elements of the covariance matrix are zero.

The combinational filter effectively computes optimal estimates of position and velocity along with the remaining elements of a state vector. The filter uses an

observation of the difference between the phase of the tracking filter and a value of the phase formulated on the time updated position of the combinational filter. Based on this measurement, the combinational filter derives estimates of position errors, velocity errors, oscillator start time and oscillator drift.

Because the Omega navigation system functions with several distinct velocity sensors (ie. inertial, doppler, or air data), a different mode of operation exists for the combinational filter for each sensor. Each method requires the filter to estimate, predict, and control a different set of system errors. Each set of system errors conforms to the dead reckoning velocity source used for time updating.

The combination of uncertainties in position error and receiver oscillator start time produces a vagueness of the number of wave lengths or lanes between the transmitting station and the receiver. This ambiguity occurs primarily when the system is first initialized. To find the solution to this lane ambiguity, several estimates of the state vector, X(i)'s, are calculated by the filter. These X(i)'s correspond to the different possible lanes or integral values of phase measurement. As measurements are acquired, corresponding to the three frequencies, the uncertainity in oscillator start time, t_o, is diminished. Measurements from different stations tend to minimize the position uncertainity. The general effect is a reduction in the number of state vector estimates, X(i)'s, that can be considered logical estimates of the most accurate state vector until only one credible estimate remains. The criterion for reasonableness is based on variance considerations.

b. Operation

There are three distinct operations identified with the combinational filter routine.

1) Initialization
2) Time update
3) Measurement update

The initialization operation assigns the initial values of variance to the covariance matrix P, sets the elements of the initial state vector to zero, and sets the system driving noises as a function of the velocity source.

In the time update operation, both the state vector X, and the covariance matrix P, are predicted from the last update. The following equations are utilized in the time update operation.

$$Xnew = \Phi new * Xold$$
$$Pnew = \Phi new * Pold * \Phi new^{T} + Nnew$$

The transition matrix Φ [6 x 6] mathematically expresses the propagation of errors across the time interval Δt since the last update. The elements of the transition matrix are shown in the following chart.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-\beta i*\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-\beta i*\Delta t \end{bmatrix}$$

where βi (i=I,D, or A) signifies the inverse correlation

time of $\delta VC2$ and $\delta VC3$ for the velocity source mode utilized. The transition matrix Φ furnishes the propagation of the predicted elements of the state vector X and the covariance matrix P across the time interval Δt. The variance effects across this time interval are propagated by the additive diagonal ncise matrix N [6 x 6]. The following chart depicts the elements of the noise matrix N.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta i*\sigma^2 i*\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta i*\sigma^2 i*\Delta t \end{bmatrix}$$

where βi is the inverse correlation time described previously, and $\sigma^2 i$ is the variance of the velocity errors ($\delta VC2$ and $\delta VC3$) and is dependent on velocity source utilized (i=I,D, or A).

During the time update operation, the multiple state vectors, X(i)'s, which exist until the correct lane has been determined must also be time updated. The time update equation for the state vector X must be sequenced through all state vectors.

$$X(i)new = \Phi new * X(i)old \qquad \text{for all } i$$

The covariance matrix time update equation is computed once, because there is only one covariance matrix regardless of the number of state vectors.

After completion of the time update operation and prior to the measurement update operation, the measurement residual, (RES), measurement matrix, M, and the measurement confidence scalar, C, are calculated for each

tracking filter input. The measurement residual and measurement matrix calculations were previously explained in the section describing the combinational filter. The measurement confidence scalar, C, is a measurement of noise computed by the following equation.

$$C = \sigma^2\phi PROP + \sigma^2\phi TRK$$

where $\sigma^2\phi PROP$ and $\sigma^2\phi TRK$ are the variance of the propagation prediction phase estimate and the tracking filter phase measurement.

The measurement update operation involves the generation of a linear, unbiased weighting vector b, and computation of a new, minimum variance, state vector X by the formula listed below.

$$Xnew = Xold + b * (RES)$$

To generate the optimum weighting vector, b, three expressions are required.

1) the predicted measurement variance - $M*P*M^T$
2) the measurement confidence scalar - C
3) the divergence control factor - ϵ (epsilon)

The weighting vector is calculated from the following equations.

$$Q = M*P*M^T + C$$

$$b = (P*M^T + \epsilon M^T) / Q$$

where Q is defined to be a scalar quantity depicting the lane variance, and ϵ is a factor to prevent divergence of the estimation process. Also updated in the measurement update process is the covariance matrix based on the

following formula.

$$Pnew = Pold - b * M * Pold + \epsilon * M^T * b^T$$

The retention of new state vectors is predicated on the testing of the position elements in the state vector, and the variance values in the covariance matrix.

7. Propagation Prediction

In order to effectively resolve the aircraft position from the phase information received, the phase velocity of the wave along the propagated path and the length of the path must be known. Several factors or effects create imperfections in the wave guide through which the VLF signals propagate. The propagation prediction function establishes the best estimate of phase and phase variance taking into account the factors listed below which effect the phase velocity of the propagated signals.

a. Diurnal Effects

The positional changes of the sun over the earth adjust the size and the shape of the ionosphere. The change of the wave guide through which the VLF signals propagate causes a change in the velocities of the phase signal which results in a difference in phase angle received at the Omega receiver location.

b. Ground Conductivity

This factor takes into account the effect of various mediums over which VLF signals travel. Water, which is a near perfect conductor of VLF signals, does not greatly affect the signals. Land, which is a less perfect conductor of VLF signals, does affect the signals to a greater extent.

Conductivity patterns have been measured and are predictable [Ref. 6].

c. Earth's Magnetic Field

This factor affects both the attenuation rate and the velocity of the signal. The phase shift on refraction of the VLF signal off of the ionosphere depends on the interaction of the signal with the electrons, and this depends on the orientation of the magnetic field with respect to the direction of propagation and on the magnitude of the field [Ref. 12].

d. Latitude and Spheroidal Effects

This factor takes into account the nonspherical shape of the earth and the adjusted path of travel from the transmitting station to the Omega receiver.

e. Polar Cap Absorption (PCA)

VLF signals which pass over the Polar Cap experience an abnormal rate of absorption during both day and night. PCA produces large changes in VLF signal patterns which can last for several days.

f. Sudden Ionospheric Disturbance (SID)

Solar flares emanating on the sun's surface increase the ionization within the atmosphere. These SID's cause large changes in the phase of VLF signals passing through the area of activity.

Of the six factors which affect the propagation of VLF signals, the first four are predictable and can be compensated for algorithmically, while the last two are unpredictable and cannot be compensated for within an Omega navigation system.

IV. VELOCITY AND NAVIGATION PROCESSING FUNCTION

The velocity and navigation processing function is divided into two routines in the Omega navigation system. The velocity processing routine, utilizing the current navigational source (ie. inertial, doppler, or air data), calculates the velocities along the system axes from the available source velocities. These system axes velocities, combined with the state vector corrections from the combinational filter, are supplied to the navigation routine in order to update the Omega system position matrix and the Omega aircraft latitude and longitude position.

A. VELOCITY PROCESSING

In the velocity processing routine, the aircraft current true airspeed velocity (VTAS) and heading (ACHDG), the current wind direction (WD) and velocity (VW), and the dopplers current velocity components along heading (VDA) and across heading (VDC) are utilized to compute the velocities in the North/South and East/West directions from the following equations.

$$VDN = VDA * COS(ACHDG) - VDC * SIN(ACHDG)$$
$$VDE = VDC * COS(ACHDG) + VDA * SIN(ACHDG)$$
$$VAN = VTAS * COS(ACHDG) + VW * COS(WD)$$
$$VAE = VTAS * SIN(ACHDG) + VW * SIN(WD)$$

where VDN and VDE signify doppler velocities North/South and East/West, and VAN and VAE signify air data velocities

North/South and East/West respectively. In constructing the data flow graph of these equations, it was self-evident that some variables could be calculated, stored in memory, and retrieved from memory when they were required in later calculations. Figure 12 depicts the data flow graph for the North/South and East/West computations. Implementing PL/M code directly from this data flow graph was straight forward.

On completion of the North/South and East/West velocities computations, a test is made in the velocity processing function to determine if the navigation mode has changed since the last iteration of the velocity function. If an upgrade of the navigation mode has occurred (ie. doppler to inertial, or air data to doppler or inertial), then the summations (DELVC2 and DELVC3) of the corrections to the system velocity components from the combinational filter are set equal to zero, and new velocities along the system axes are calculated. Figure 13 depicts the navigation mode test and applicable equations for computing the system axes velocities.

In order to save execution time in the calculation of the formulas and to simplify the PL/M code, the functions

```
(ViE * COS(SHDGA) + ViN * SIN(SHDGA))
(ViN * COS(SHDGA) - ViE * SIN(SHDGA))
```

were calculated prior to the navigation mode test. Figure 14 depicts the data flow graph for these two functions. After successful computation of the above functions and the navigation mode test, the corrected velocities (VC2 and VC3) along the system axes are calculated and the resultant values used in the navigation routine.

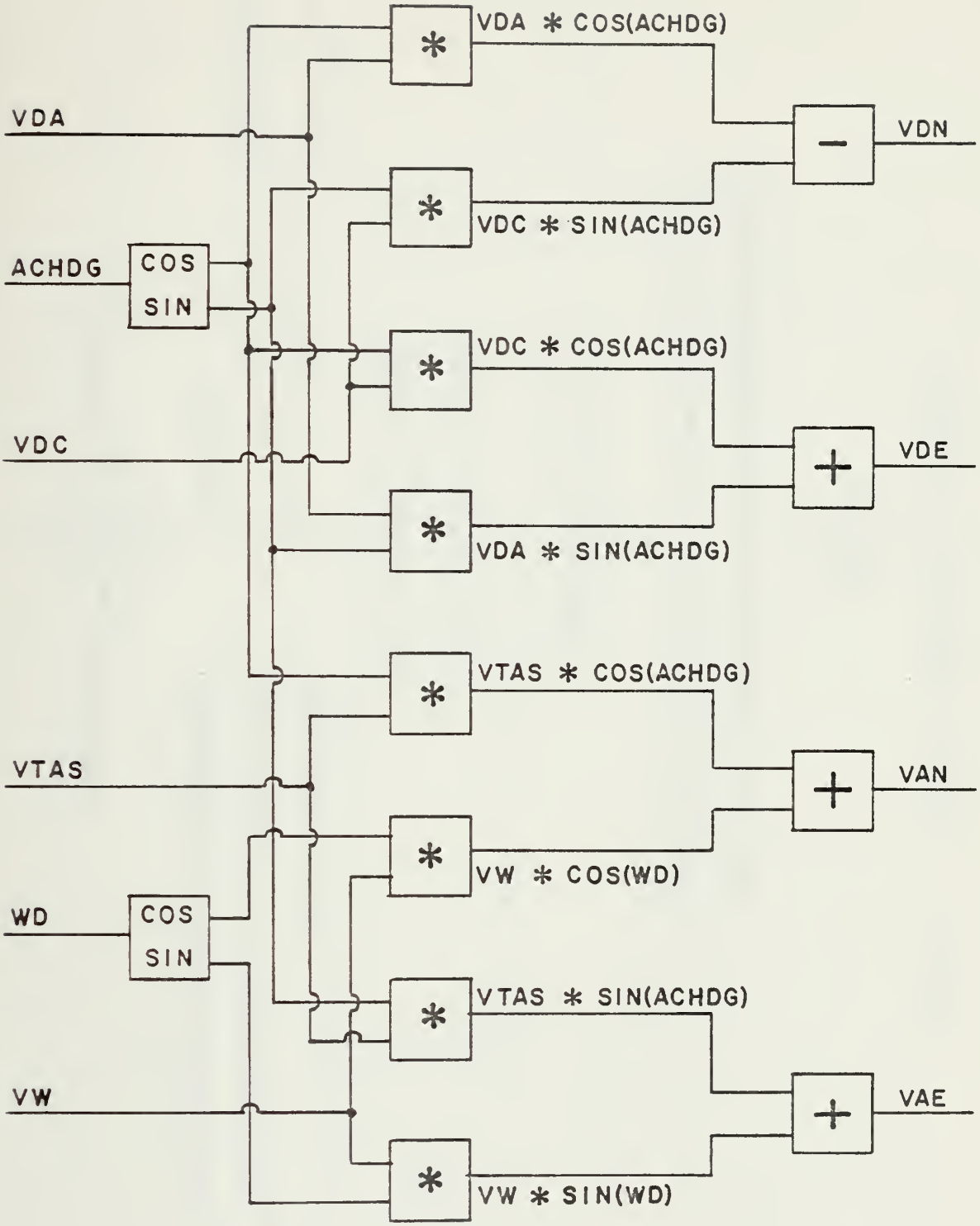


FIGURE 12 - NORTH/SOUTH/EAST/WEST VELOCITY DATA FLOW GRAPH

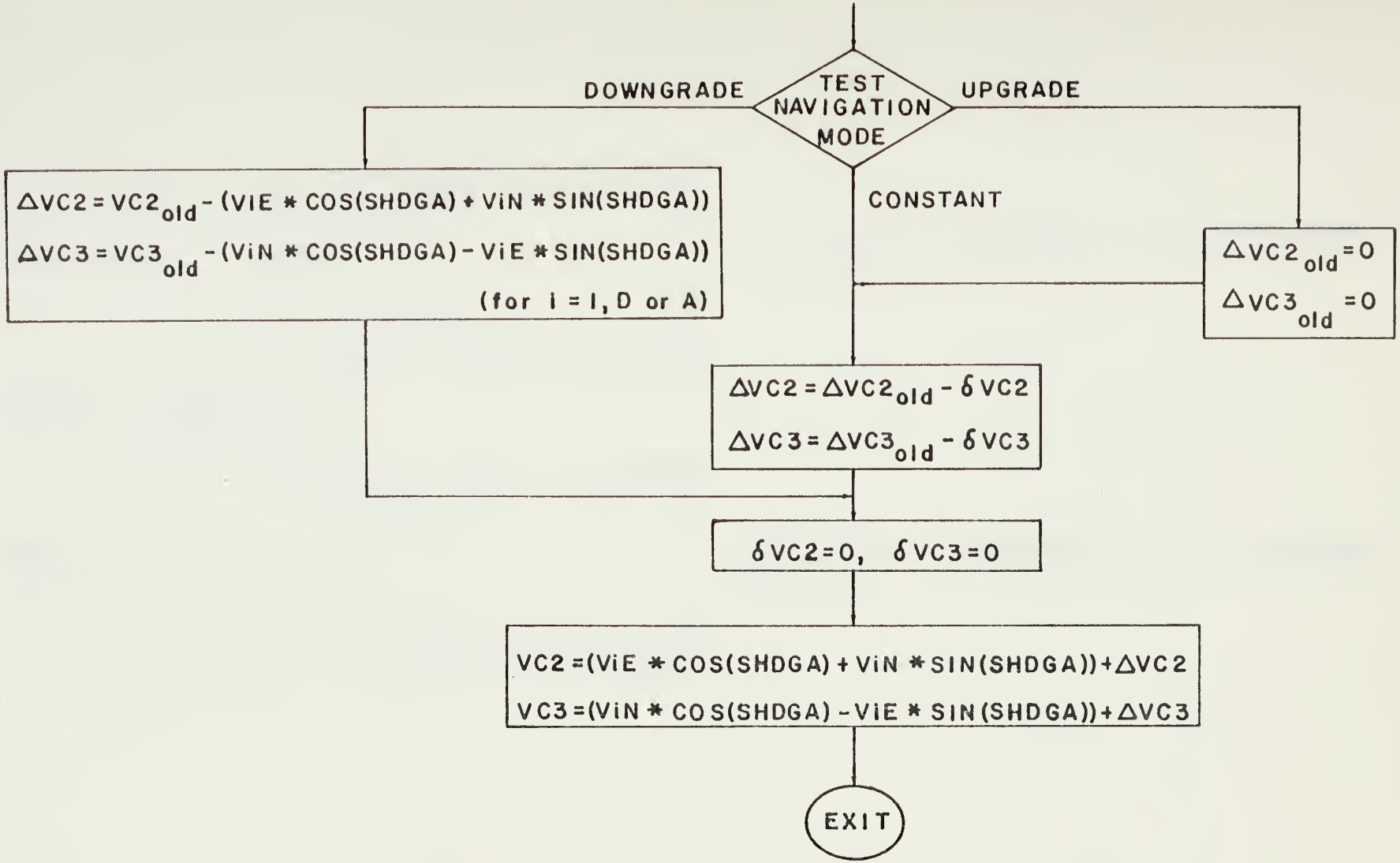


FIGURE 13 - VELOCITY PROCESSING FLOWCHART

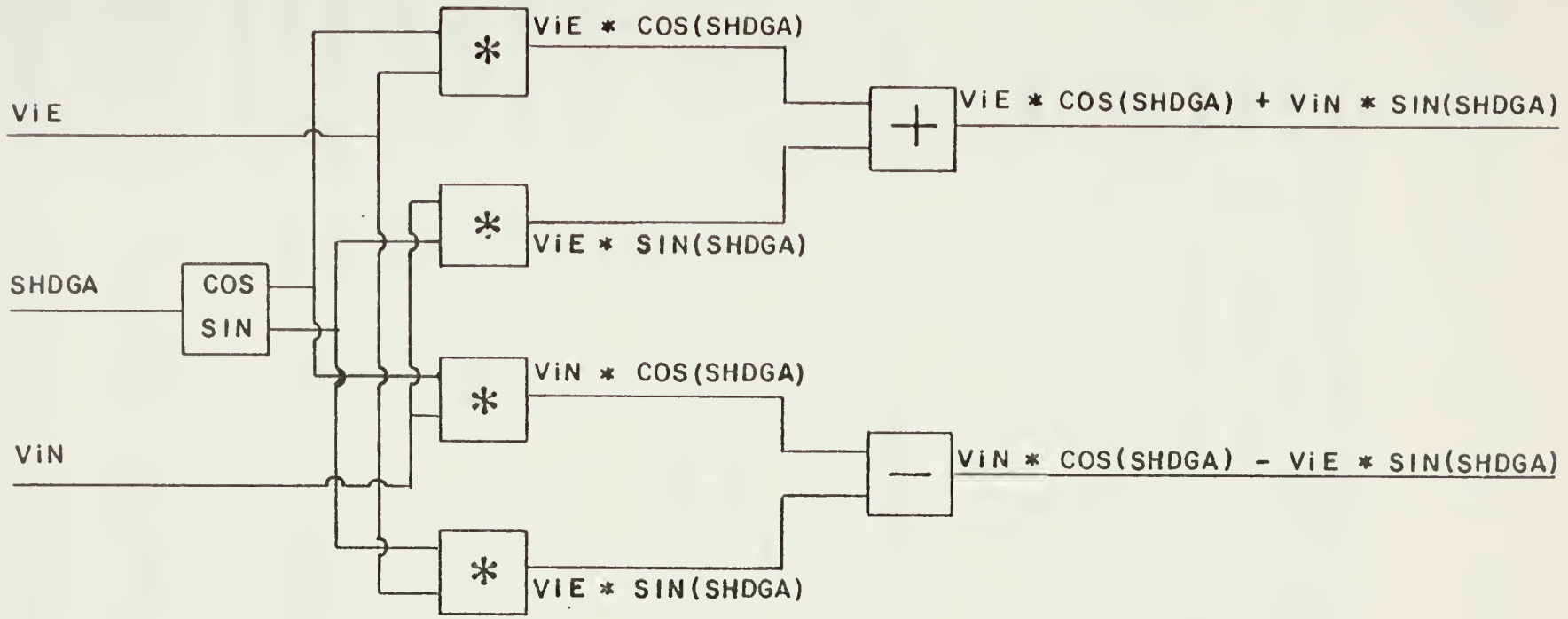


FIGURE 14 - PARTIAL FUNCTION DATA FLOW GRAPH

B. NAVIGATION PROCESSING

In the navigation processing routine, the Omega system position matrix R [3 X 3] is initialized, the angular rotations (DELT2 and DELT3) about the system axes are calculated using the corrected velocities supplied by the velocity processing routine and the state vector corrections from the combinational filter, the system position matrix is updated, and a new Omega aircraft latitude (OLATAC) and longitude (OLONAC) are calculated. The flowchart of the navigation routine is shown in Figure 15.

The initialization of the Omega system position matrix R occurs during the initialization function when the Navigator/Communicator inserts the aircraft initial latitude (OLATIN) and longitude (OLONIN). These values, with an initial condition that the system heading angle (SHDGA) is zero, are input to the equations listed below.

R11 = SIN(OLATIN)

R12 = COS(OLATIN) * COS(OLONIN)

R13 = COS(OLATIN) * SIN(OLONIN)

R21 = COS(OLATIN) * SIN(SHDGA) = 0

R22 = -SIN(OLONIN) * COS(SHDGA) -
SIN(OLATIN) * COS(OLONIN) * SIN(SHDGA)
= -SIN(CLONIN)

R23 = COS(OLONIN) * COS(SHDGA) -
SIN(CLATIN) * SIN(OLONIN) * SIN(SHDGA)
= COS(CLONIN)

R31 = COS(OLATIN) * COS(SHDGA) = COS(OLATIN)

```
R32 =  SIN(OLONIN) * SIN(SHDGA) -
       SIN(OLATIN) * COS(OLONIN) * COS(SHDGA)
    = -SIN(CIATIN) * COS(OLONIN)

R33 = -COS(CLONIN) * SIN(SHDGA) -
       SIN(CIATIN) * SIN(OLONIN) * COS(SHDGA)
    = -SIN(OLATIN) * SIN(OLONIN)
```

These nine equations represent nine direction cosine values. These values represent the orientation of each of the three system axes (R1, R2, and R3) in the earth's fixed coordinate frame. In constructing the data flow graph for the initialization equations, the concept of minimizing execution time prevailed, and as variables were calculated, they were stored in memory and retrieved as they were required in later calculations. Figure 16 portrays the data flow graph for the initialization of the position matrix R.

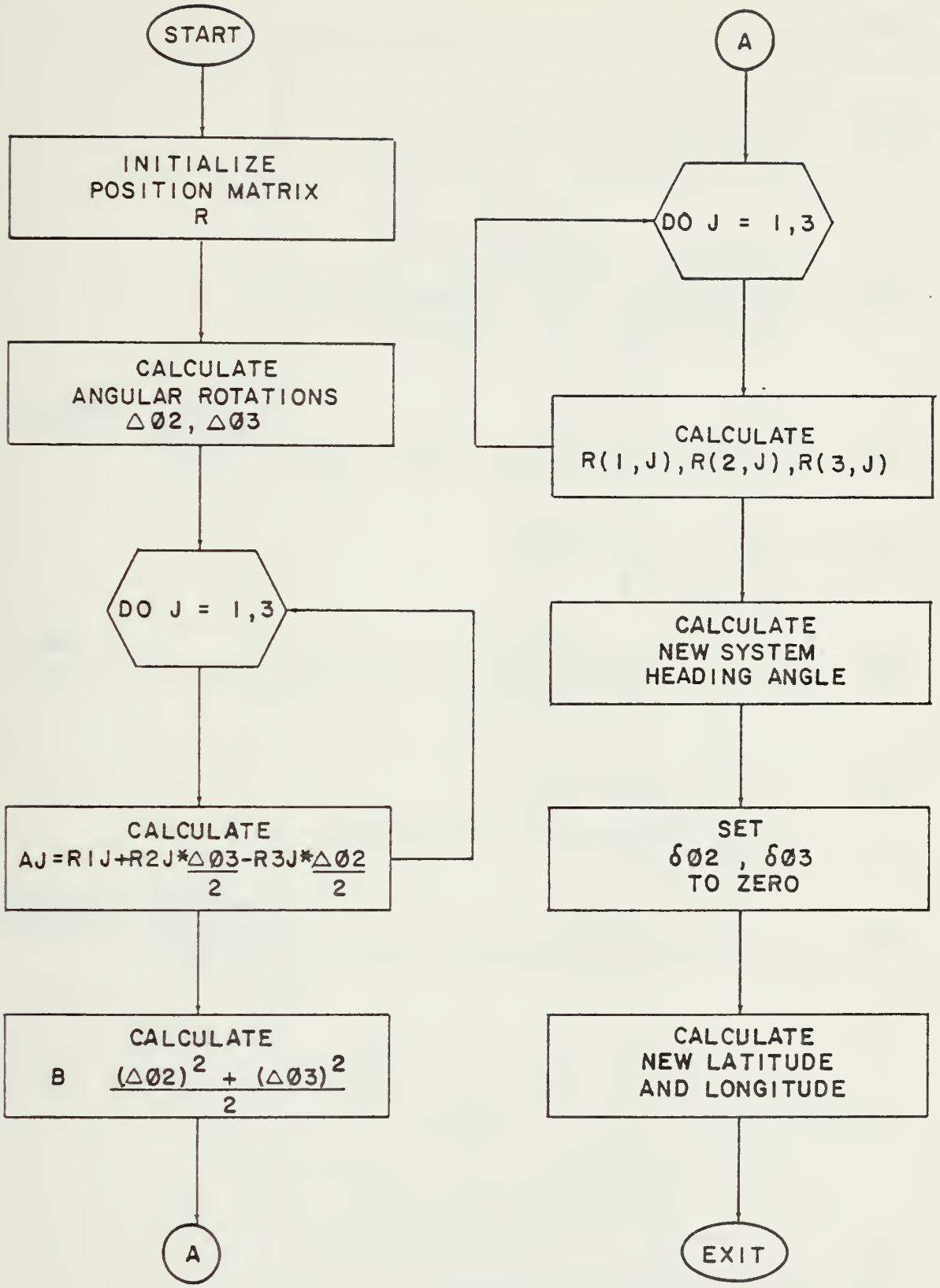


FIGURE 15 - NAVIGATION PROCESSING FLOWCHART

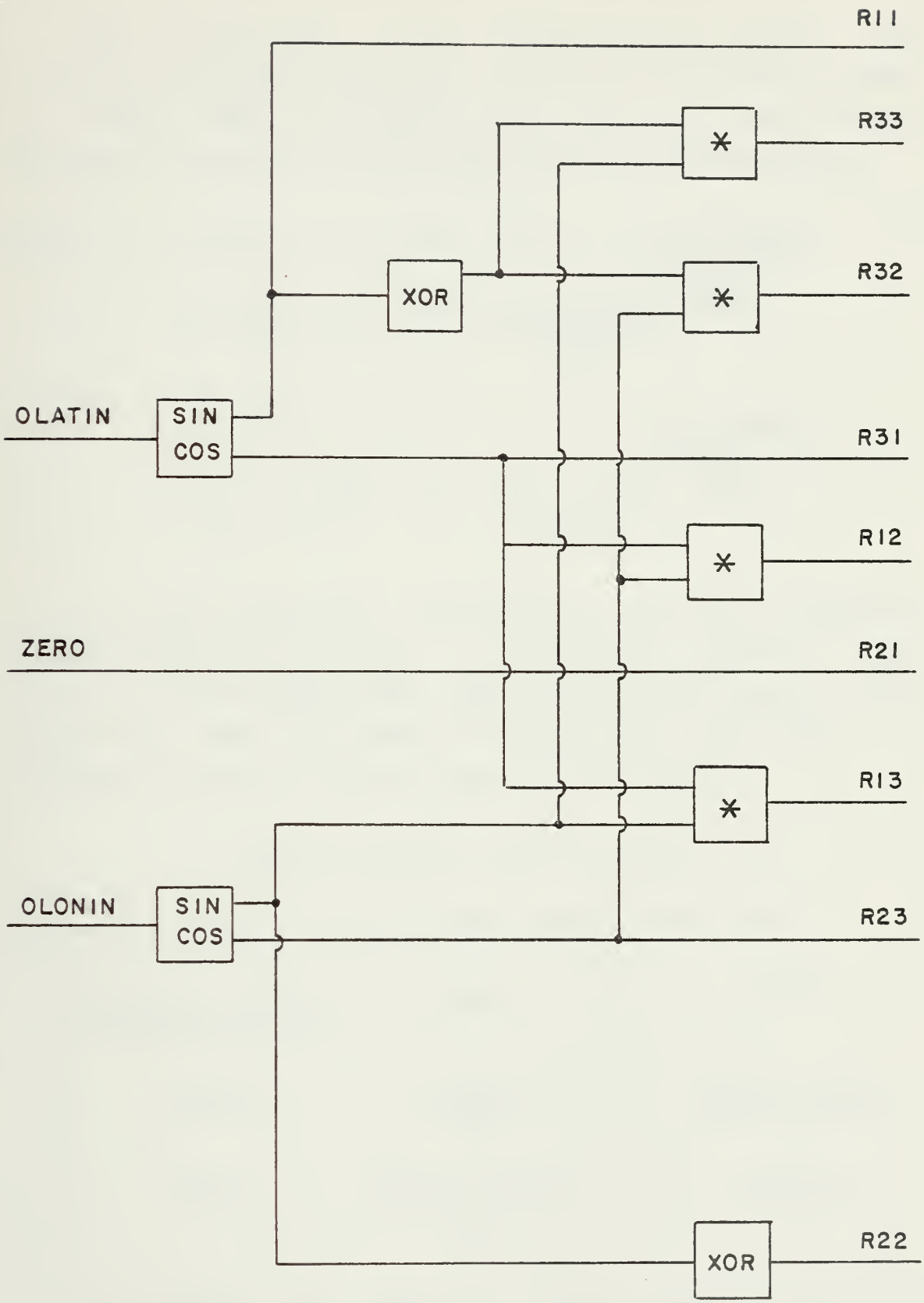


FIGURE 16 - INITIALIZATION OF POSITION MATRIX R

After completing the initialization phase, the navigation function calculates the angular rotations (DELT2 and DELT3) about the system axes R2 and R3. The following equations are used in calculating the angular rotations.

$$DELT2 = -[1 + EEC * (2 * R31^2 - R11^2)] * \frac{VC3 * DTNAV}{EMER} - 2 * EEC * R21 * R31 * \frac{VC2 * DTNAV}{EMER} + \delta\phi 3$$

$$DELT3 = [1 + EEC * (2 * R21^2 - R11^2)] * \frac{VC2 * DTNAV}{EMER} + 2 * EEC * R21 * R31 * \frac{VC3 * DTNAV}{EMER} - \delta\phi 2$$

As in previous equations, converting to data flow graphs was accomplished with the idea of minimizing execution time. Figure 17 is the data flow graph constructed from the above equations. Once the angular rotations are calculated, the position matrix is updated based on the formula

$$[R]new = [R]old + [\phi] * [R]old$$

where $[\phi]$ is the rotation update matrix given below.

$$\begin{bmatrix} -\frac{(DELT2^2 + DELT3^2)}{2} & DELT3 & -DELT2 \\ -DELT3 & -\frac{(DELT3)}{2} & \frac{DELT2 * DELT3}{2} \\ DELT2 & \frac{DELT2 * DELT3}{2} & -\frac{(DELT2)}{2} \end{bmatrix}$$

Simplification of the position matrix update formula is accomplished utilizing the following equations.

$$A(j) = R1j + R2j * (DELT3 / 2) - R3j * (DELT2 / 2)$$

$$B = [\,(DELT2)^2 + (DELT3^2)\,] / 2$$

The position matrix update equations are then written in the following format.

$$R1j = R1j - R1j * B + R2j * DELT3 - R3j * DELT2$$

$$R2j = R2j - DELT3 * A(j)$$

$$R3j = R3j + DELT2 * A(j)$$

Implementation of these equations in a data flow graph is depicted in Figure 18.

Once the position matrix R is updated, the last equations to calculate are for updating the system heading angle and the aircraft new latitude and longitude. The equations listed below update these three system variables.

$$SHDGA = ARCTAN\ (R21 / R31)$$

$$OLATAC = ARCTAN\ [\,(R11 * COS(SHDGA)) / R31\,]$$

$$OLONAC = ARCTAN\ (R13 / R12)$$

The data flow graph presented in Figure 19 depicts the update of these system parameters.

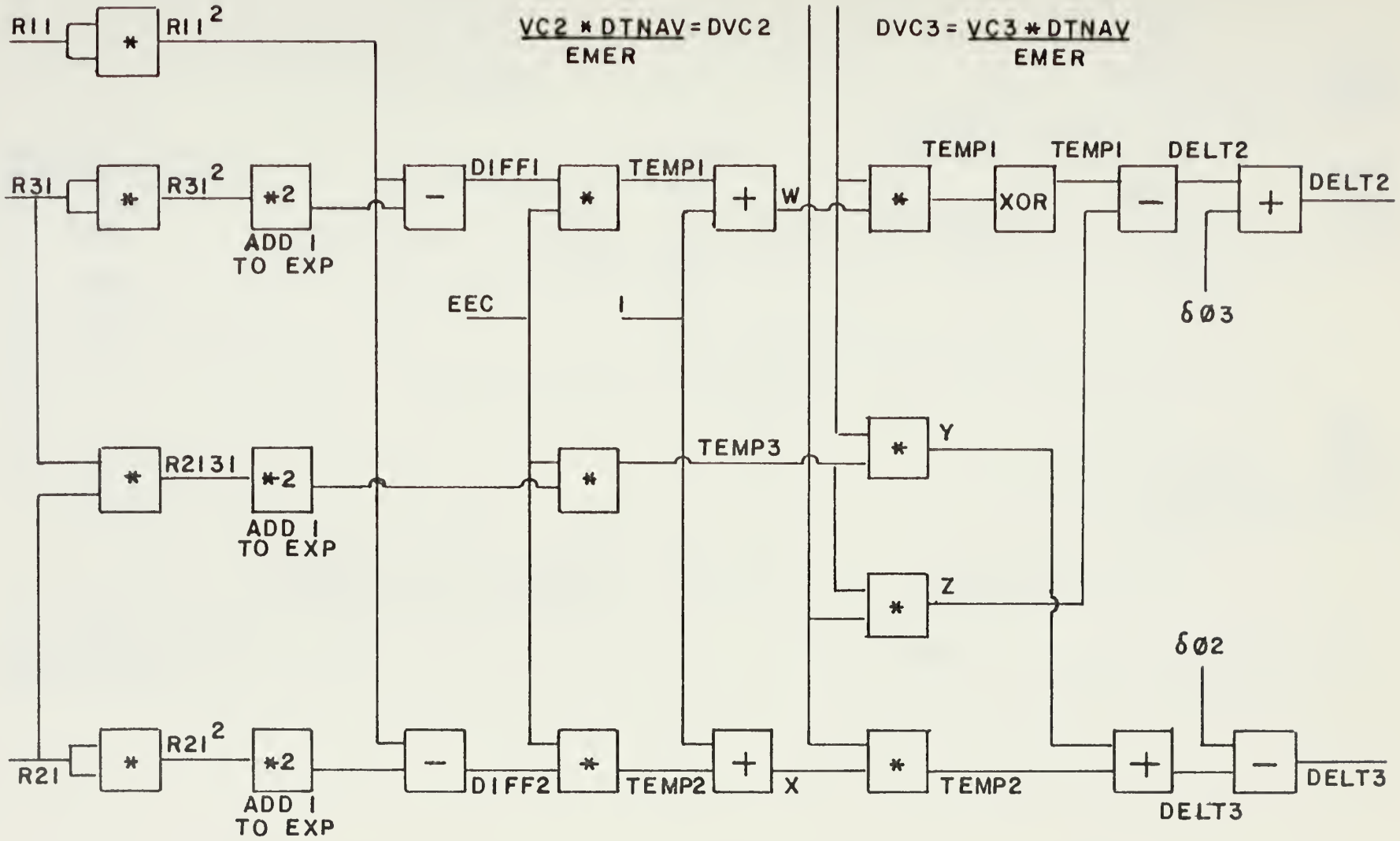


FIGURE 17 - ANGULAR ROTATIONS DATA FLOW GRAPH

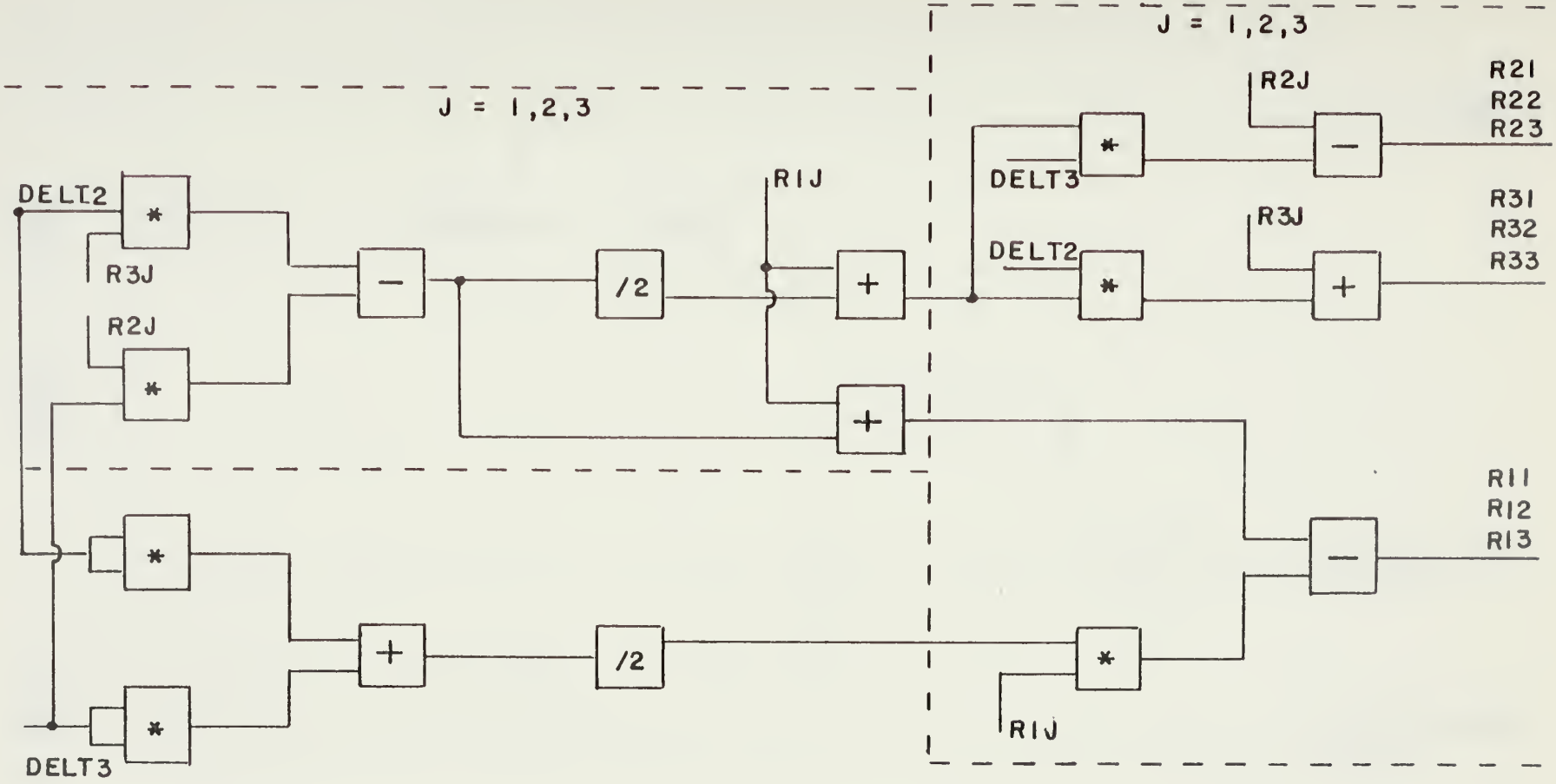


FIGURE 18 – UPDATE OF POSITION MATRIX DATA FLOW GRAPH

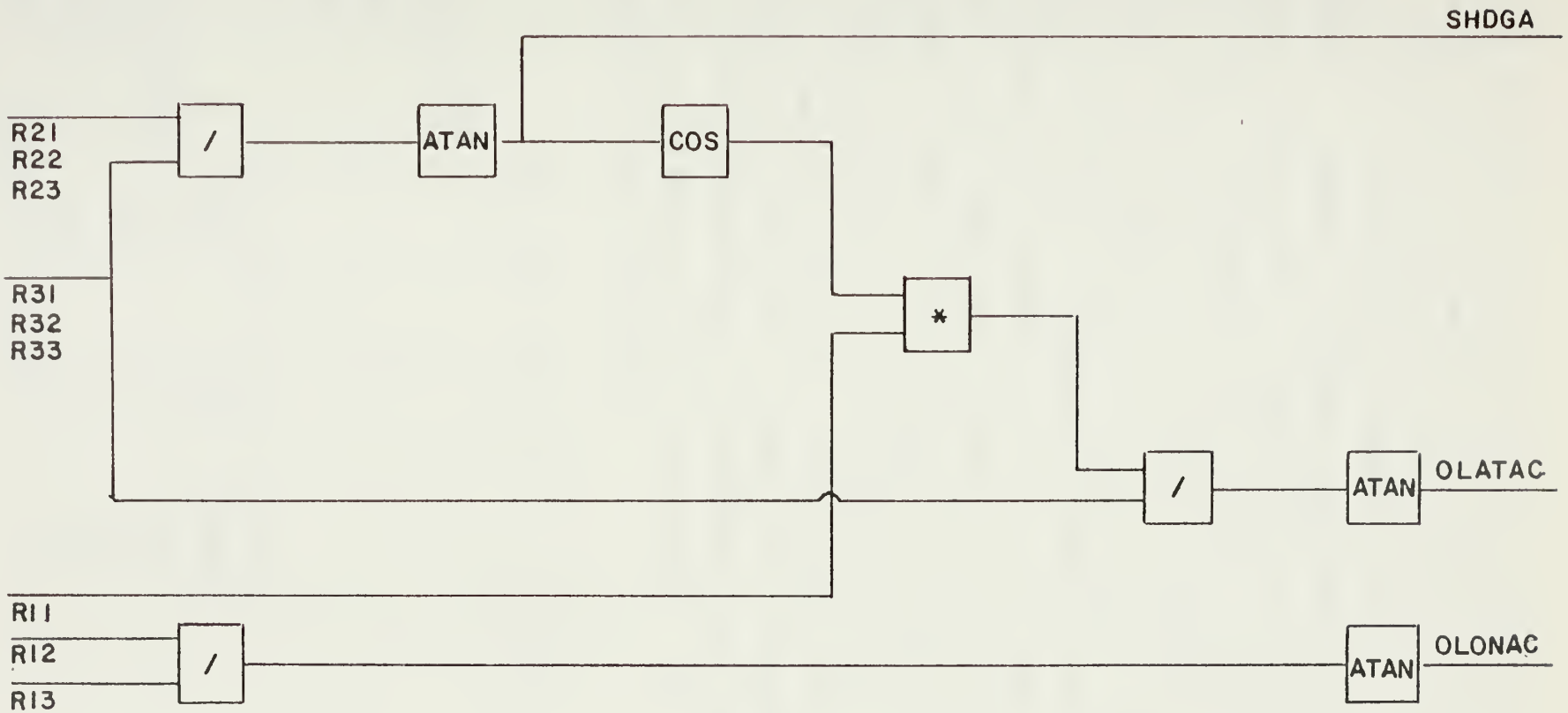


FIGURE 19 – UPDATE SYSTEM HEADING ANGLE, LATITUDE, LONGITUDE

From the data flow graphs of the velocity and navigation function, four PL/M floating point subroutines were written. The subroutines consist of an initialization (INITIALIZE$R) routine for initializing the position matrix R, a velocity processing (VELPROC) routine for computing the velocities along the system axes, a navigation processing (NAVPROC) routine for calculating the angular rotations about the system axes, and a new position (NEWPOS) routine for updating the position matrix R, aircraft system heading angle, and aircraft latitude and longitude. Based on the arithmetic and/or trigonometric functions used in the data flow graphs and an approximation of the time required to execute each function, a preliminary estimate of the time required to execute each subroutine was acquired. Utilizing INTERP/80, which is the FORTRAN IV software simulation of the INTEL 8080 CPU and is available on the IBM 360/67, the actual execution time of each program was acquired. The following chart shows the number of arithmetic functions used in each subroutine, the estimated and actual execution times in milliseconds of the subroutines, and the storage utilized for each subroutine.

	ADD	SUB	MULT	DIV	COSSIN	ATAN
	(1.0-1.2)		(1.5-2.0)		(55.0-66.0)	(35.0-40.0)
INITIALIZE$R	0	0	4	0	2	0
VELPROC	6	6	12	0	3	0
NAVPROC	3	3	11	0	0	0
NEWPOS	10	9	18	3	1	3

	ESTIMATED TIME	ACTUAL TIME	STORAGE (BYTES)
INITIALIZE$R	123.0	113.4	375
VELPROC	206.7	248.4	830
NAVPROC	25.9	26.3	384
NEWPOS	227.7	287.8	842

From the last three subroutines, it is seen that the total execution time of 562.5 milliseconds is between two and one-half and three times as great as the 200 millisecond time interval currently implemented in the P-3C. However, because the Omega navigation display for Omega latitude and longitude is updated every second, the microcomputer subroutines are capable of providing an update in sufficient time to be displayed on the Omega tableau.

In the simulated test case used to check the validity of the subroutines, a navigation cycle time (DTNAV) of 1 second was used in the computations. This 1 second time interval allowed an aircraft traveling at 300 knots to be updated at approximately every 506 feet. The velocity processing routine was able to calculate the velocities along the system axes, using the PL/M floating point function, to four digit accuracy (ie. from 000.0 feet per second to 999.9 feet per second). These velocities encompass the complete range of the P-3C airspeed capabilities.

The calculations performed in the remaining subroutines presented problems when using the PL/M floating point functions. Because the functions utilize a 16 bit mantissa, the exponent values have to be within a range of 2^{16} when adding or subtracting two numbers. Because the difference between the calculated angular rotations about the system axes and the direction cosine values in the position matrix is not within this range, several angular rotations calculated over time had to be added together before updating of the position matrix R could be accomplished. This is the major difficulty with the subroutines implemented with the present PL/M floating point routines.

V. CONCLUSION

Based on the subroutines implemented for the velocity and navigation function, a microcomputer is capable of calculating the required system parameters in sufficient time and with sufficient accuracy to provide reliable position fixing information. Even with the microcomputers primary disadvantage of slower execution time, the position fixing information could be displayed at one second intervals provided a floating point package which operated with a mantissa of 24 bits could be acquired. Further research of the remaining internal functions and the overall effect they have on the velocity and navigation function is encouraged. Particular attention should be focused on the state vector elements applied to the velocity and navigation equations, and their relevance to overall system operation and update of position information.

```
DECLARE ZZ ADDRESS, ZE BYTE;

 /*
 THE ABOVE VARIABLES ARE GLOBAL
 TO THE FLOATING POINT SUBROUTINES
 */


ERROR: PROCEDURE(I);

 /*
 PROCEDURE TO PROVIDE ERROR MESSAGES
 FOLLOWED BY TERMINATION OF PROGRAM EXECUTION
 */

DECLARE I BYTE;
   CALL CRLF;

   DO CASE I;

      /*
      MESSAGE FOR UNDERFLOW
      */
      CALL PRINT(.'EXECUTION TERMINATED DUE TO UNDERFLOW$');

      /*
      MESSAGE FOR OVERFLOW
      */
      CALL PRINT(.'EXECUTION TERMINATED DUE TO OVERFLOW$');

      /*
      MESSAGE FOR ERROR IN LOG-ARGUMENT
      */
      CALL PRINT(.'ARGUMENT FOR LOG NEGATIVE OR ZERO$');

      /*
      MESSAGE FOR ERROR IN DIVISION
      */
      CALL PRINT(.'ATTEMPTED DIVISION BY ZERO$');

      /*
      MESSAGE FOR INPUT ERROR
      */
      CALL PRINT(.'INPUT ERROR$');

      /*
      MESSAGE FOR EXP(X) OVERFLOW
      */
      CALL PRINT(.'OVERFLOW, ARGUMENT FOR EXP(X) TOO BIG$');

      END;

      HALT;
      RETURN;
END ERROR;


 /*
 FLOATING POINT ADD ROUTINE
 */


ADD: PROCEDURE (XA,YA,PA);
DECLARE  XA ADDRESS, X BASED XA BYTE,
         YA ADDRESS, Y BASED YA BYTE,
         PA ADDRESS, P BASED PA BYTE,
         (DIFF,I,SIGN,XEX,YEX) BYTE,
         (XX,YY) ADDRESS;
DECLARE  POSITIV LITERALLY '< 80H',
         NEGATIV LITERALLY '=80H';
```

```
/*
PROCEDURE TO LEFT JUSTIFY MANTISSA, DECREMENTING EXPONENT
*/

  ADJUST: PROCEDURE;
     DO I=1 TO 16;
        IF (ZZ AND 8000H)=8000H THEN RETURN;
        ZZ = SHL(ZZ,1);
        P(2) = (ZE := P(2) - 1);

        /*
        CHECK FOR UNDERFLOW
        */

        IF (P(2) AND 07FH) = 0 THEN
           CALL ERROR(0);
        END;

        RETURN;
  END ADJUST;

/*
BRING BOTH MANTISSAS AND EXPONENTS IN WORKING AREA
*/

  XX = SHL(DOUBLE(X),8) OR X(1);
  YY = SHL(DOUBLE(Y),8) OR Y(1);
  XEX = X(2) AND 07FH;
  YEX = Y(2) AND 07FH;

  /*
  CHECK MANTISSAS TO SEE IF ZERO
  */

  IF (XX = 0000H) THEN DO;
     ZZ   = YY;
     P(2) = (ZE := Y(2));
     GO TO RET;
     END;

  IF (YY = 0000H) THEN DO;
     ZZ   = XX;
     P(2) = (ZE := X(2));
     GO TO RET;
     END;

  /*
  CALCULATE DIFFERENCE BETWEEN EXPONENTS AND CHECK SIGN
  */
  DIFF = XEX - YEX;
  IF (DIFF AND 080H) <> 0 THEN
     DIFF = - DIFF;
  SIGN = (X(2) AND 080H) XOR (Y(2) AND 080H);

  /*
  CHECK IF OPERATION BEYOND SIGNIFICANCE
  */
  IF (DIFF >= 16) THEN DO;
     IF (XEX > YEX) THEN DO;
        ZZ = XX;
        P    = HIGH(ZZ);
        P(1) = LOW(ZZ);
        P(2) = (ZE := XEX);
        END;
     ELSE DO;
        ZZ = YY;
        P    = HIGH(ZZ);
        P(1) = LOW(ZZ);
        P(2) = (ZE := YEX);
        END;
     RETURN;
     END;
```

```
IF XEX = YEX THEN DO;

/*
EXPONENTS ARE EQUAL
*/

   IF XX > YY THEN DO;

   /*
   OPERAND 1 > OPERAND 2
   */

      P(2) = (ZE := X(2));
      IF SIGN POSITIV THEN
         GO TO EXIT1;
      GO TO EXIT2;
      END;

   IF YY > XX THEN DO;

   /*
   OPERAND 2 > OPERAND 1
   */

      P(2) = (ZE := Y(2));
      IF SIGN POSITIV THEN
         GO TO EXIT1;
      GO TO EXIT3;
      END;

   /*
   OPERAND 1 = OPERAND 2
   */

   IF SIGN POSITIV THEN DO;
      P(2) = (ZE := X(2));
      GO TO EXIT1;
      END;

   /*
   RESULTING ZERO FROM EQUAL NUMBERS WITH DIFFERENT SIGN
   */

   ZZ = XX - YY;
   P    = HIGH(ZZ);
   P(1) = LOW(ZZ);
   P(2) = (ZE := 000H);
   RETURN;
   END;

IF XEX > YEX THEN DO;

/*
EXPONENT OF OPERAND 1 > EXPONENT OF OPERAND 2
*/
   P(2) = (ZE := X(2));
   YY = SHR(YY,DIFF);
   IF SIGN POSITIV THEN
      GO TO EXIT1;
   GO TO EXIT2;
   END;

/*
EXPONENT OF OPERAND 2 > EXPONENT OF OPERAND 1
*/

P(2) = (ZE := Y(2));
XX = SHR(XX,DIFF);
IF SIGN POSITIV THEN
   GO TO EXIT1;
ELSE
   GO TO EXIT3;
```

```
    /*
    ADD TWO OPERANDS WITH EQUAL SIGN
    */

    EXIT1:
    ZZ = XX + YY;
    IF (CARRY) THEN DO;
       ZZ = SCR(ZZ,1);
       ZE = ZE + 1;

       /*
       ADJUST EXPONENT AND CHECK FOR OVERFLOW
       */

       IF (P(2) POSITIV) THEN DO;
          P(2) = (ZE := P(2) + 1);
          IF (P(2) AND 080H) <> 0 THEN
             CALL ERROR(1);
          END;
       ELSE DO;
          P(2) = (ZE := P(2) + 1);
          IF (CARRY) THEN
             CALL ERROR(1);
          END;
       END;

    GO TO RET;

    /*
    ADD OPERANDS WITH DIFFERENT SIGN
    */

    EXIT2:
    ZZ = XX - YY;
    CALL ADJUST;
    GO TO RET;

    EXIT3:
    ZZ = YY - XX;
    CALL ADJUST;

    /*
    ASSIGN RESULTING MANTISSA VALUES
    */

    RET:
    P    = HIGH(ZZ);
    P(1) = LOW(ZZ);
    RETURN;
END ADD;


 /*
 FLOATING POINT SUBTRACT ROUTINE
 */

SUB: PROCEDURE (XA,YA,PA);
DECLARE XA ADDRESS, X BASED XA BYTE,
        YA ADDRESS, Y BASED YA BYTE,
        PA ADDRESS, P BASED PA BYTE,
        NEGY (3) BYTE;

    /*
    CHANGE SIGN OF OPERAND 2 AND CALL ADD ROUTINE
    */

    NEGY    = Y;
    NEGY(1) = Y(1);
    NEGY(2) = Y(2) XOR 080H;
    CALL ADD(XA,.NEGY,PA);
END SUB;
```

```
 /*
 FLOATING POINT MULTIPLY ROUTINE INCLUDING
 OPTION FOR HARDWARE MULTIPLY OF MANTISSA
 */

DECLARE (MD$SLOT,MD$OPTION) BYTE INITIAL (0);

MULT: PROCEDURE(XA,YA,PA);
DECLARE XA ADDRESS, X BASED XA BYTE,
        YA ADDRESS, Y BASED YA BYTE,
        PA ADDRESS, P BASED PA BYTE,
        I BYTE, (OP1, OP2) ADDRESS;

   /*
   CHECK FOR ZERO OPERAND
   */

   IF (X=0) OR (Y=0) THEN DO;
   /*

   IF FOUND, RETURN ZERO AS RESULT
   */
      P(0) = 000H;
      P(1) = 000H;
      P(2) = 000H;
      RETURN;
      END;

   /*
   HARDWARE OR SOFTWARE MANTISSA
   */

   DO CASE MD$OPTION;

   /*
   CASE 0 - SOFTWARE MULTIPLY
   */

      DO;
      /*
      BRING BOTH MANTISSAS IN OPERAND FIELDS
      */

      ZE   = SHR( X(1), 1);
      OP1  = SHL(DOUBLE(X),7);
      OP1  = OP1 OR ZE;
      OP2  = SHL(DOUBLE(Y),8);
      OP2  = OP2 OR Y(1);

      /*
      THE MULTIPLICATION OF THE MANTISSAS IS DONE IN
      HARDWARE-LIKE FASHION BY SHIFTING, COPYING AND
      ADDING OF OPERAND 1 FOR EVERY 1 FOUND IN OPER-
      AND 2; AND BY SHIFTING ONLY FOR EVERY 0 FOUND
      IN OPERAND 2, STARTING WITH THE LEAST SIGNIFI-
      CANT 1
      */

      /*
      INITIALIZE WORKING FIELD
      */

      ZZ = 0;
      ZE = 0;

      /*
      SHIFT OPERAND 2 BIT BY BIT THROUGH CARRY AFTER
      SHIFTING OF ADDED RESULTS IN WORKING FIELD HAS
      BEEN DONE
      */
```

```
    DO I=0 TO 15 BY 1;
       ZZ  = SCR(ZZ,1);
       OP2 = SCR(OP2,1);

       /*
       CHECK IF A 1 HAS BEEN SHIFTED INTO THE CARRY
       */

       IF (CARRY) THEN DO;

       /*
       IF YES, ADD OPERAND 1 TO THE CONTENTS OF THE
       WORKING AREA
       */

           ZZ = ZZ + OP1;
           END;
        END;

    /*
    NORMALIZE, IF NEEDED
    */

    IF (ZZ < 8000H) THEN DO;
       ZZ = SHL(ZZ,1);

       /*
       SET ADJUSTMENT FOR EXPONENT
       */

       ZE = -1;
       END;
    END;    /* CASE 0 */

/*
CASE 1 - HARDWARE MULTIPLY, SLOT 0
*/

    DO;
    OUTPUT(0) = X(1);
    OUTPUT(1) = X;
    OUTPUT(2) = Y(1);
    OUTPUT(3) = Y;
    ZZ = SHL(DOUBLE(INPUT(3)),8) + DOUBLE(INPUT(2));
    IF (ZZ AND 8000H) = 0 THEN DO;
       I = SCL(INPUT(1),1);
       ZZ = SCL(ZZ,1);
       ZE = -1;
       END;
    ELSE
       ZE = 0;
    END;    /* CASE 1 */

/*
CASE 2 - HARDWARE MULTIPLY, SLOT 1
*/

    DO;
    OUTPUT(04H) = X(1);
    OUTPUT(05H) = X;
    OUTPUT(06H) = Y(1);
    OUTPUT(07H) = Y;
    ZZ = SHL(DOUBLE(INPUT(07H)),8) + DOUBLE(INPUT(06H));
    IF (ZZ AND 8000H) = 0 THEN DO;
       I = SCL(INPUT(05H),1);
       ZZ = SCL(ZZ,1);
       ZE = -1;
       END;
    ELSE
       ZE = 0;
    END;    /* CASE 2 */
```

```
   /*
   CASE 3 - HARDWARE MULTIPLY, SLOT 2
   */

      DO;
      OUTPUT(08H) = X(1);
      OUTPUT(09H) = X;
      OUTPUT(0AH) = Y(1);
      OUTPUT(0BH) = Y;
      ZZ = SHL(DOUBLE(INPUT(0BH)),8) + DOUBLE(INPUT(0AH));
      IF (ZZ AND 8000H) = 0 THEN DO;
         I = SCL(INPUT(09H),1);
         ZZ = SCL(ZZ,1);
         ZE = -1;
         END;
      ELSE
         ZE = 0;
      END;    /* CASE 3 */


   /*
   CASE 4 - HARDWARE MULTIPLY, SLOT 3
   */


      DO;
      OUTPUT(0CH) = X(1);
      OUTPUT(0DH) = X;
      OUTPUT(0EH) = Y(1);
      OUTPUT(0FH) = Y;
      ZZ = SHL(DOUBLE(INPUT(0FH)),8) + DOUBLE(INPUT(0EH));
      IF (ZZ AND 8000H) = 0 THEN DO;
         I = SCL(INPUT(0DH),1);
         ZZ = SCL(ZZ,1);
         ZE = -1;
         END;
      ELSE
         ZE = 0;
      END;    /* CASE 4 */
   END;       /* CASE I STATEMENT */

   /*
   SUM UP EXPONENTS AND ADJUSTMENT
   */

   P(1) = (X(2) AND 7FH) + (Y(2) AND 7FH) ;
   P(2) = P(1) + ZE - 64;

   /*
   CHECK FOR OVERFLOW AND UNDERFLOW
   */

   IF (CARRY) THEN
      CALL ERROR(C);
   IF (P(2) AND 080H) <> 0 THEN
      CALL ERROR(1);

   /*
   SET SIGN BIT
   */

   P(2) = P(2) OR ((X(2) AND 080H) XOR (Y(2) AND C80H));

   /*
   ASSIGN MANTISSA VALUES
   */

   P    = HIGH(ZZ);
   P(1) =  LOW(ZZ);

   RETURN;
END MULT;
```

```
/*
FLOATING POINT DIVIDE ROUTINE INCLUDING DIVIDE BY HARDWARE
*/

DIV: PROCEDURE(XA,YA,PA);
DECLARE XA ADDRESS, X BASED XA BYTE,
        YA ADDRESS, Y BASED YA BYTE,
        PA ADDRESS, P BASED PA BYTE,
        (OP1, OP2, DIFF) ADDRESS,
        I BYTE, C (3) BYTE;

   /*
   BRING BOTH MANTISSAS IN OPERAND FIELD
   */

   OP1 = SHL(DOUBLE(X),8) OR X(1);
   OP2 = SHL(DOUBLE(Y),8) OR Y(1);

   /*
   CHECK FOR ZERO IN OPERAND 1
   */

   IF X = 0 THEN DO;

   /*
   IF FOUND RETURN ZERO AS RESULT
   */

      P(0) = 000H;
      P(1) = 000H;
      P(2) = 000H;
      RETURN;
      END;

   /*
   CHECK FOR ZERO IN OPERAND 2
   */

   IF Y = 0 THEN
      CALL ERROR(3);

   /*
   0 = SOFTWARE DIVIDE, 1-4 = HARDWARE DIVIDE
   */

   DO CASE MO$OPTION;

   /*
   CASE 0 - SOFTWARE DIVIDE
   */

      DO;

      /*
      BEGIN DIVISION BY SUBTRACTING BOTH OPERANDS RESP.
      THE DIFFERENCE AND SHIFTING/ADDING BITS TO THE RE-
      SULTING MANTISSA FOR EACH POSSIBLE SUBTRACTION WITH
      POSITIVE RESULT, AND SHIFTING ONLY AND ADDING ONE
      PLACE TO OPERAND 1 IN CASE SUBTRACTION YIELDED A
      NEGATIVE RESULT
      */

      OP1 = SHR(OP1,1);
      OP2 = SHR(OP2,1);
      ZZ = 0;
      ZE = 0;
      I = 1;
      DIFF = OP1 - OP2;
```

```
IF NOT CARRY THEN DO;
   IF DIFF = 0 THEN DO;

   /*
   RESULT FOR NUMBER DIVIDED BY ITSELF
   */
      ZZ = 8000H;
      GO TO SETX;
      END;
   GO TO NOCARRY;
   END;

OP1 = SHL(OP1,1);
ZE = -1;

NEXTSTEP:
DIFF = OP1 - OP2;
IF CARRY THEN DO;
   OP1 = SHL(OP1,1);
   GO TO SHIFT;
   END;

/*
NO CARRY: SET OPERAND 1 TO SHIFTED DIFFERENCE,
ADD ONE TO RESULTING MANTISSA
*/

NOCARRY:
OP1 = SHL(DIFF,1);
ZZ = ZZ + 1;

/*
SHIFT MANTISSA ONE PLACE TO THE LEFT
*/

SHIFT:
ZZ = SHL(ZZ,1);
I = I + 1;

/*
REPEAT UNTIL NORMALIZED
*/
IF I < 16 THEN
   GO TO NEXTSTEP;

/*
SET EXPONENT
*/

SETX:
ZE = (X(2) AND 07FH) - (Y(2) AND 07FH) + ZE + 65;

/*
CHECK FOR OVERFLOW AND UNDERFLOW
*/
IF (ZE AND 07FH) < 0 THEN
   CALL ERROR(0);
IF (ZE AND 080H) <> 0 THEN
   CALL ERROR(1);

/*
SET SIGN BIT
*/
P(2) = ZE OR ((X(2) AND 080H) XOR (Y(2) AND 080H));

/*
ASSIGN RESULTING MANTISSA VALUES
*/
P    = HIGH(ZZ);
P(1) = LOW(ZZ);
RETURN;
END;   /* CASE 3 */
```

```
/*
CASE 1 - HARDWARE DIVIDE, SLOT 0
*/

   DO;
   OUTPUT(30H) = Y(1);
   OUTPUT(31H) = Y;
   OUTPUT(33H) = 0;
   C = INPUT(31H);
   C(1) = INPUT(30H);
   END;    /* CASE 1 */

/*
CASE 2 - HARDWARE DIVIDE, SLOT 1
*/

   DO;
   OUTPUT(34H) = Y(1);
   OUTPUT(35H) = Y;
   OUTPUT(37H) = 0;
   C = INPUT(35H);
   C(1) = INPUT(34H);
   END;    /* CASE 2 */

/*
CASE 3 - HARDWARE DIVIDE, SLOT 2
*/

   DO;
   OUTPUT(38H) = Y(1);
   OUTPUT(39H) = Y;
   OUTPUT(3BH) = 0;
   C = INPUT(39H);
   C(1) = INPUT(38H);
   END;    /* CASE 3 */

/*
CASE 4 - HARDWARE DIVIDE, SLOT 3
*/

   DO;
   OUTPUT(3CH) = Y(1);
   OUTPUT(3DH) = Y;
   OUTPUT(3FH) = 0;
   C = INPUT(3DH);
   C(1) = INPUT(3CH);
   END;    /* CASE 4 */
END;       /* CASE I STATEMENT */

/*
MULTIPLY RECIPROCAL OF Y BY X TO GET
RESULT FOR HARDWARE CASES
*/

IF (Y(2) AND 7FH) < 2 THEN

/*
OVERFLOW PROTECTION
*/
   DO;
   C(2) = (Y(2) AND 80H) OR (7FH - (Y(2) AND 7FH));
   CALL MULT(XA,.C,PA);
   IF ((P(2) AND 7FH) + 2) > 7FH THEN

   /*
   OVERFLOW
   */

      CALL ERROR(1);
   ELSE
      P(2) = P(2) + 2;
   END;
```

```
   ELSE

   /*
   NO OVERFLOW PROTECTION NEEDED
   */

      DO;
      C(2) = (Y(2) AND 80H) OR (81H - (Y(2) AND 7FH));
      CALL MULT(XA,.C,PA);
      END;

   RETURN;
END DIV;


SQRT: PROCEDURE (XA,PA);
DECLARE XA ADDRESS, X BASED XA BYTE,
        PA ADDRESS, P BASED PA BYTE,
        (C,C1,C2,C3,B,B1,B2,R) BYTE;

 /*
 ASSUME THAT XA IS A POSITIVE REAL NUMBER AND THE
 INITIAL APPROXIMATION FOR THE ROOT IS MANT*EXP/2
 */

   B  = X;
   B1 = X(1);
   B2 = X(2) - 64;
   IF (B2 AND 080H) = 0 THEN
      B2 = SHR(B2,1) + 64;
   ELSE
      DO;
      B2 = -B2;
      B2 = SHR(B2,1);
      B2 = 64 - B2;
      END;

   DO R=1 TO 4;
      CALL DIV(XA,.B,.C);
      CALL ADD(.C,.B,.B);
      B(2)=B(2)-1;
   END;

   P  = B;
   P(1)= B(1);
   P(2)= B(2);
   RETURN;
END SQRT;

DECLARE (ID,TEMP,TEMP1,TEMP2,ENTRY,ENTRY1,ENTRY2) BYTE;


COMPARE: PROCEDURE (XA,YA) BYTE;
DECLARE XA ADDRESS, X BASED XA BYTE,
        YA ADDRESS, Y BASED YA BYTE,
        (CHECK1, CHECK2) ADDRESS,
        (XP,YP,XE,YE) BYTE;

   /*
   MOVE EXPONENTS INTO WORKING AREA
   */

   XE=X(2) AND 80H;
   YE=Y(2) AND 80H;
   IF (XE = YE) THEN DO;

   /*
   EXPONENTS EQUAL
   */

      XP=X(2) AND 7FH;
      YP=Y(2) AND 7FH;
```

```
      /*
      MOVE MANTISSAS INTO WORKING AREA
      */

      CHECK1=SHL(DOUBLE(X),8) OR X(1);
      CHECK2=SHL(DOUBLE(Y),8) OR Y(1);
      IF (XE = 080H) THEN DO;
         IF (XP < YP) THEN
            RETURN 2;
         IF (XP > YP) THEN
            RETURN 0;
         IF CHECK2 < CHECK1 THEN
            RETURN 0;
         IF CHECK2 > CHECK1 THEN
            RETURN 2;
         RETURN 1;
         END;
      IF (XP < YP) THEN
         RETURN 0;
      IF (XP > YP) THEN
         RETURN 2;
      IF CHECK2 < CHECK1 THEN
         RETURN 2;
      IF CHECK2 > CHECK1 THEN
         RETURN 0;
      RETURN 1;
      END;
   IF (XE = 0) THEN
      RETURN 2;
   RETURN 0;
END COMPARE;

DECLARE (T,T1,T2,Z,Z1,Z2) BYTE;

FUNCTION: PROCEDURE(I,XA);
DECLARE I BYTE, XA ADDRESS, X BASED XA BYTE;
 /*
 I=0  ATAN = ATANGENT
 I=1  IS NOT USED PRESENTLY
 I=2  COSSIN = COSINE AND SINE
 */

 /*
 COS: PROCEDURE (XA);
 */
DECLARE TEMPCOS (195) BYTE INITIAL(
0CCH,CC0H,0C0H,0C9H,00EH,03BH,0C9H,00EH,03CH,096H,0CAH,03DH,
0C9H,0CEH,03DH,
0FBH,C51H,C3DH,096H,0CAH,03EH,0AFH,0ECH,03EH,0C9H,00EH,03EH,
0E2H,02FH,C3EH,
0FBH,051H,C3EH,08AH,039H,03FH,096H,0CAH,C3FH,0A3H,05BH,03FH,
0AFH,CECH,03FH,
0BCH,C7DH,03FH,0C9H,00EH,03FH,0D5H,09EH,03FH,0E2H,02FH,03FH,
0EEH,0C0H,03FH,
0FBH,C51H,C3FH,083H,0F1H,040H,08AH,039H,040H,090H,082H,040H,
096H,0CAH,040H,
09DH,012H,040H,0A3H,05BH,040H,0A9H,0A3H,040H,0AFH,0ECH,040H,
0B6H,034H,040H,
0BCH,C7DH,040H,0C2H,0C5H,040H,0C9H,00EH,040H,0CFH,056H,040H,
0D5H,C9EH,C40H,
0DBH,CE7H,040H,0E2H,02FH,040H,0E8H,078H,040H,0EEH,0C0H,040H,
0F5H,0C9H,040H,
0FBH,C51H,C40H,C8CH,0CCH,041H,083H,0F1H,041H,087H,015H,041H,
08AH,C39H,041H,
08DH,C5DH,041H,090H,082H,041H,093H,0A6H,041H,096H,0CAH,041H,
099H,0EEH,041H,
09DH,012H,041H,0A0H,037H,041H,0A3H,05BH,041H,0A6H,07FH,041H,
0A9H,0A3H,041H,
0ACH,0C8H,041H,0AFH,0ECH,041H,0B3H,010H,041H,0B6H,034H,041H,
0B9H,C58H,041H,
0BCH,C7DH,041H,0BFH,0A1H,041H,0C2H,0C5H,041H,0C5H,0E9H,041H,
0C9H,0CFH,041H);
```

```
DECLARE VALCOS (195) BYTE INITIAL(

080H,00CH,041H,0FFH,0ECH,040H,0FFH,0B1H,040H,0FFH,04EH,040H,
0FEH,0C4H,040H,
0FEH,013H,040H,0FDH,03AH,040H,0FCH,03BH,040H,0FBH,014H,040H,
0F9H,0C7H,040H,
0F8H,C54H,040H,0F6H,0BAH,040H,0F4H,0FAH,040H,0F3H,014H,040H,
0F1H,C09H,040H,
0EEH,0D8H,040H,0ECH,083H,040H,0EAH,0DAH,040H,0E7H,06CH,040H,
0E4H,CAAH,040H,
0E1H,0C6H,040H,0DEH,0BEH,040H,0DBH,094H,040H,0D8H,049H,040H,
0D4H,0D8H,040H,
0D1H,04EH,040H,0CDH,09FH,040H,0C9H,0D2H,040H,0C5H,0E5H,040H,
0C1H,0D9H,040H,
0BDH,0B0H,040H,0B9H,069H,040H,0B5H,006H,040H,0B0H,087H,040H,
0ABH,0ECH,040H,
0A7H,037H,040H,0A2H,069H,040H,09DH,081H,040H,098H,081H,040H,
093H,C6AH,C40H,
08EH,038H,040H,088H,0F7H,040H,083H,09EH,040H,0FCH,061H,03FH,
0F1H,05FH,03EH,
0E6H,C3EH,03FH,0DAH,0EDH,03FH,0CFH,080H,03FH,0C3H,0F4H,03FH,
0B8H,C49H,03FH,
0ACH,082H,03FH,0A0H,0A0H,03FH,094H,0A5H,03FH,088H,094H,03FH,
0F8H,0DBH,03EH,
0E0H,C68H,03EH,0C7H,0D2H,03EH,0AFH,01DH,03EH,096H,04DH,03EH,
0FAH,0CDH,03DH,
0C8H,0D8H,03DH,096H,0C5H,03DH,0C9H,034H,03CH,0C9H,07FH,03BH,
000H,00CH,000H);



DECLARE DIFFCOS (195) BYTE INITIAL (

0C9H,CC5H,0BAH,096H,0C1H,0BCH,0FBH,027H,0BCH,0AFH,0B3H,0BDH,
0E1H,CB8H,0BDH,
089H,0CDH,0BEH,0A2H,0A8H,0BEH,0BBH,06BH,0BEH,0D4H,011H,0BEH,
0ECH,097H,0BEH,
082H,078H,0BFH,08EH,098H,0BFH,09AH,09EH,0BFH,0A6H,08CH,0BFH,
0B2H,C61H,0BFH,
0BEH,01AH,0BFH,0C9H,0B6H,0BFH,0D5H,033H,0BFH,0E0H,08EH,0BFH,
0EBH,0C8H,0BFH,
0F6H,0DDH,0BFH,080H,0E5H,0C0H,086H,049H,0C0H,08BH,098H,0C0H,
C90H,0D1H,0C0H,
095H,0F4H,0C0H,09BH,000H,0C0H,09FH,0F4H,0C0H,0A4H,0CFH,0C0H,
0A9H,091H,0C0H,
0AEH,039H,0C0H,0B2H,0C6H,0C0H,0B7H,037H,0C0H,0BBH,08CH,0C0H,
0BFH,0C4H,0C0H,
0C3H,0DFH,0C0H,0C7H,0DBH,0C0H,0CBH,0B9H,0C0H,0CFH,077H,0C0H,
0D3H,015H,0C0H,
0D6H,093H,0C0H,0D9H,0EFH,0C0H,0DDH,02AH,0C0H,0E0H,043H,0C0H,
0E3H,C39H,0C0H,
0E6H,00CH,0C0H,0E8H,0BCH,0C0H,0EBH,048H,0C0H,0EDH,0AFH,0C0H,
0EFH,0F2H,0C0H,
0F2H,010H,0C0H,0F4H,009H,0C0H,0F5H,0DCH,0C0H,0F7H,089H,0C0H,
0F9H,01CH,0C0H,
0FAH,07CH,0C0H,0FBH,0AAH,0C0H,0FCH,0BDH,0C0H,0FDH,0A9H,0C0H,
0FEH,06EH,0C0H,
0FFH,C0CH,0C0H,0FFH,082H,0C0H,0FFH,0D1H,0C0H,0FFH,0F9H,0C0H,
0CCH,0CCH,0C0H);
```

```
 /*
 ATAN1: PROCEDURE (XA);
 */
DECLARE TEMPATAN (249) BYTE INITIAL(
000H,C0CH,000H,080H,000H,03BH,080H,000H,03CH,0C0H,000H,03CH,
C8CH,CCCH,03DH,
0A0H,0C0H,03DH,0C0H,000H,03DH,0E0H,000H,03DH,080H,000H,03EH,
090H,00CH,03EH,
0A0H,0C0H,03EH,0B0H,000H,03EH,0C0H,000H,03EH,0D0H,000H,03EH,
0ECH,00CH,03EH,
0F0H,000H,03EH,080H,000H,03FH,088H,000H,03FH,090H,000H,03FH,
098H,000H,03FH,
0A0H,000H,03FH,0A8H,000H,03FH,0B0H,000H,03FH,0B8H,000H,03FH,
0C0H,CCCH,03FH,
0C8H,000H,03FH,0D0H,000H,03FH,0D8H,000H,03FH,0ECH,000H,03FH,
0E8H,00CH,03FH,
0F0H,C00H,03FH,0F8H,000H,03FH,080H,000H,040H,084H,000H,040H,
088H,00CH,040H,
08CH,000H,040H,090H,000H,040H,094H,000H,040H,098H,000H,040H,
09CH,00CH,040H,
0A0H,0C0H,040H,0A4H,000H,040H,0A8H,000H,040H,0ACH,000H,040H,
0BCH,0CCH,040H,
0B4H,0C0H,040H,0B8H,000H,040H,0BCH,000H,040H,0C0H,000H,040H,
0C4H,000H,040H,
0C8H,C0CH,040H,0CCH,000H,040H,0D0H,000H,040H,0D4H,000H,040H,
0D8H,00CH,040H,
0E0H,000H,040H,0E8H,000H,040H,0F0H,000H,040H,0F8H,000H,040H,
080H,0C0H,041H,
082H,C70H,041H,084H,0E1H,041H,087H,052H,041H,089H,0C3H,041H,
08EH,CA5H,041H,
093H,C87H,041H,098H,069H,041H,09DH,04BH,041H,0A2H,02DH,041H,
0A7H,00FH,041H,
0ABH,0FCH,041H,0B0H,0D2H,041H,0B5H,0B4H,041H,0BAH,096H,041H,
0BFH,C78H,041H,
0C4H,C5AH,041H,0C9H,03CH,041H,0CEH,01EH,041H,0D2H,0FFH,041H,
0D7H,0E1H,041H,CDCH,0C3H,041H,0E1H,0A5H,041H,0E6H,087H,041H)


DECLARE VALATAN(249) BYTE INITIAL(
000H,00CH,000H,0FFH,0FAH,03AH,0FFH,0EAH,03BH,0BFH,0DCH,03CH,
0FFH,0AAH,03CH,
09FH,0ACH,03DH,0BFH,070H,03DH,0DFH,01CH,03DH,0FEH,0ADH,03DH,
08FH,0CFH,03EH,
09EH,0B7H,03EH,0AEH,04CH,03EH,0BDH,0CBH,03EH,0CDH,035H,03EH,
0DCH,086H,03EH,
0EBH,0BEH,03EH,0FAH,0DBH,03EH,084H,0EEH,03FH,08CH,05FH,03FH,
093H,0C1H,03FH,
09BH,013H,03FH,0A2H,055H,03FH,0A9H,085H,03FH,0B0H,0A4H,03FH,
0B7H,0B0H,03FH,
0BEH,0ABH,03FH,0C5H,092H,03FH,0CCH,066H,03FH,0D3H,027H,03FH,
0D9H,0D4H,03FH,
0E0H,06DH,03FH,0E6H,0F2H,03FH,0EDH,063H,03FH,0F3H,0BFH,03FH,
0FAH,006H,03FH,
080H,01CH,040H,083H,02BH,040H,086H,030H,040H,089H,02AH,040H,
08CH,01AH,040H,
08FH,0C0H,040H,091H,0DBH,040H,094H,0ACH,040H,097H,073H,040H,
09AH,02FH,040H,
09CH,0E1H,040H,C9FH,089H,040H,0A2H,028H,C40H,0A4H,0BCH,040H,
0A7H,C46H,040H,
0A9H,CC7H,040H,0ACH,03EH,040H,0AEH,0ACH,040H,0B1H,010H,040H,
0B3H,C6BH,040H,
0B8H,005H,040H,0BCH,07BH,040H,0C0H,0CEH,040H,0C4H,0FFH,040H,
0C9H,00FH,040H,
0CBH,07AH,040H,0CDH,0DAH,040H,0DDH,02EH,040H,0D2H,076H,040H,
0D6H,0E7H,040H,
0DBH,02DH,040H,0DFH,04AH,040H,0E3H,041H,040H,0E7H,013H,040H,
0EAH,0C1H,040H,
0EEH,04CH,040H,0F1H,0B8H,040H,0F5H,004H,040H,0F8H,032H,040H,
0FBH,044H,040H,
0FEH,03BH,040H,08CH,08CH,041H,081H,0EDH,041H,083H,043H,041H,
084H,C8EH,041H,085H,0CEH,041H,087H,003H,041H,088H,02FH,041H)
```

```
DECLARE DIFFATAN (249) BYTE INITIAL(
0FFH,0FAH,040H,0FFH,0DAH,040H,0FFH,09AH,040H,0FFH,03BH,040H,
0FEH,08CH,040H,
0FEH,01EH,040H,0FDH,061H,040H,0FCH,087H,040H,0FBH,08EH,040H,
0FAH,07AH,040H,
0F9H,049H,040H,0F7H,0FDH,040H,0F6H,096H,040H,0F5H,017H,040H,
0F3H,07FH,040H,
0F1H,0C0H,040H,0F0H,0DAH,040H,0EEH,030H,040H,0ECH,041H,040H,
0EAH,03FH,040H,
0E8H,02CH,040H,0E6H,009H,040H,0E3H,0D6H,040H,0E1H,095H,040H,
0DFH,047H,040H,
0DCH,0ECH,040H,0DAH,088H,040H,0D8H,019H,040H,0D5H,0A2H,040H,
0D3H,024H,040H,
0D0H,09EH,040H,0CEH,013H,040H,0CBH,084H,040H,0C8H,0F1H,040H,
0C6H,058H,040H,
0C3H,0C4H,040H,0C1H,02BH,040H,0BEH,092H,040H,0BBH,0FAH,040H,
0B9H,062H,040H,
0B6H,0CCH,040H,0B4H,038H,040H,0B1H,0A8H,040H,0AFH,018H,040H,
0ACH,092H,040H,
0AAH,00DH,040H,0A7H,08DH,040H,0A5H,012H,040H,0A2H,09DH,040H,
0A0H,02EH,040H,
09DH,0C5H,040H,09BH,063H,040H,099H,007H,040H,096H,0B2H,040H,
093H,041H,040H,
08EH,0C4H,040H,08AH,065H,040H,086H,025H,040H,082H,005H,040H,
0FDH,092H,03FH,
0F8H,0C8H,03FH,0F4H,016H,03FH,0EFH,07CH,03FH,0E8H,0C3H,03FH,
0E0H,01AH,03EH,
0D7H,0CCH,03FH,0CFH,0D7H,03FH,0C8H,038H,03FH,0C0H,0EEH,03FH,
0B9H,0F3H,03FH,
0B3H,048H,03FH,0ACH,0E7H,03FH,0A6H,0CDH,03FH,0A0H,0F8H,03FH,
098H,065H,03FH,
096H,00EH,03FH,090H,0F9H,03FH,08CH,018H,03FH,087H,070H,03FH,
082H,0FBH,03FH,0FDH,06AH,03EH,0F5H,045H,03EH,0EDH,076H,03EH)

DECLARE (E,V,D,V1,D1) (3) ADDRESS ,
        (LOOKI, MAXI) BYTE,
        LOOK (3) BYTE INITIAL (249,0,195),
        MAX (3) BYTE INITIAL (249,0,195),
        ENTA ADDRESS, ENT BASED ENTA BYTE,
        VALA ADDRESS, VAL BASED VALA BYTE,
        DIFFA ADDRESS, DIFF BASED DIFFA BYTE,
        VAL1A ADDRESS, VAL1 BASED VAL1A BYTE,
        DIFF1A ADDRESS, DIFF1 BASED DIFF1A BYTE;

 /*
 INPUT CONSISTS OF A BASED VARIABLE, OUTPUT WILL BE
   IN GLOBAL VARIABLES T, T1, T2,  AND Z, Z1, Z2
 */

   E(0) = .TEMPATAN;
   V(0) = .VALATAN;
   D(0) = .DIFFATAN;
   E(2) = .TEMPCOS;
   V(2) = .VALCOS;
   D(2) = .DIFFCOS;

   ENTA = E(I);
   VALA = V(I);
   DIFFA = D(I);
   VAL1A = V1(I);
   DIFF1A = D1(I);
   LOOKI = LOOK(I);
   MAXI = MAX(I);
   ENTRY = X;
   ENTRY1 = X(1);
   ENTRY2 = X(2);
   IF (LOOKI = MAXI) THEN
      LOOKI = 0;
   ID=COMPARE(.ENTRY,ENTA+LOOKI);
   IF (ID = 1) THEN
      GO TO XIT1;
```

```
    /*
    READ UPWARDS IN THE TABLE
    */

    IF (ID > 1) THEN DO;
LP1:  IF (LOCKI = MAXI) THEN
         GO TO XIT1;
      LOCKI=LOCKI+3;
      ID=COMPARE(.ENTRY,ENTA+LOOKI);
      IF (ID = 1) THEN
         GO TO XIT1;
      IF (ID > 1) THEN
         GO TO LP1;
      LOCKI=LOCKI-3;
      GO TO XIT2;
      END;

    /*
    READ DOWNWARDS IN THE TABLE
    */

LP2: IF (LOCKI = 0) THEN
        GO TO XIT1;
     LOCKI=LOCKI-3;
     ID=COMPARE(.ENTRY,ENTA+LOOKI);
     IF (ID = 1) THEN
        GO TO XIT1;
     IF (ID < 1) THEN
        GO TO LP2;
     GO TO XIT2;

XIT1: T  = VAL(LOCKI);
      T1 = VAL(LOCKI+1);
      T2 = VAL(LOCKI+2);
      IF (I <> 1) THEN
         GO TO RET;
      Z  = VAL1(LOOKI);
      Z1 = VAL1(LOOKI+1);
      Z2 = VAL1(LOOKI+2);
      GO TO RET;

XIT2: TEMP  = ENT(LOOKI);
      TEMP1 = ENT(LOOKI+1);
      TEMP2 = ENT(LOOKI+2);

      /*
      CHANGE THE SIGN OF TEMP
      */
      TEMP2 = TEMP2 XOR 80H;

      /*
      TEMP = ENTRY - ENT(LOOK)

      */
      CALL ADD(.TEMP,.ENTRY,.TEMP);

      /*
      (ENTRY - ENT(LOOK))*DIFF(LOOK)
      */
      CALL MULT(.TEMP,DIFFA+LOOKI,.T);
      CALL ADD(.T,VALA+LOOKI,.T);

      /*
      VAL(LOOK) + ENTRY - ENT(LOOK) * DIFF(LOOK)
      */
      IF (I <> 1) THEN
         GO TO RET;
      CALL MULT(.TEMP,DIFF1A+LOOKI,.Z);
      CALL ADD(.Z,VAL1A+LOOKI,.Z);
RET:  LOCK(I) = LOCKI;
      RETURN;
END FUNCTION;
```

```
CCSSIN: PROCEDURE(XA,AC,AS);
DECLARE XA ADDRESS, X BASED XA BYTE,
        AC ADDRESS, S BASED AC BYTE,
        AS ADDRESS, Y BASED AS BYTE,
        (I,TH,TH1,TH2,THN,THN1,THN2) BYTE,
        (MPI2,MPI21,MPI22) BYTE INITIAL (0C9H,0FH,0C1H);
DECLARE OUT LABEL;

 /*
 THIS ROUTINE COMPUTES BOTH THE COSINE AND THE SINE
 FOR THE GIVEN ANGLE XA IN RADIANS.  THE OUTPUT VALUES
 ARE STORED IN THE GLOBAL VARIABLES Z, Z1, AND Z2 FOR
 COSINE, AND T, T1, AND T2 FOR SINE
 */
   TH  = X;
   TH1 = X(1);
   TH2 = X(2);
   DO I=1 TO 4;
      THN  = TH;
      THN1 = TH1;
      THN2 = TH2;

      /*
      THETA = THETA - (PI/2)
      */
      CALL ADD (.TH,.MPI2,.TH);

      /*
      THN = THETA BAR AND TH = THETA BAR - (PI/2)
      */
      IF (((ZE AND 80H) <> 0) OR ((ZZ = 0000H) AND
                                  (ZE = 00H))) THEN
         GO TO OUT;
      END;
   RETURN;

OUT:
   TH2=ZE AND 7FH;
   CALL FUNCTION (2,.THN);
   THN  = T;
   THN1 = T1;
   THN2 = T2;

   /*
   THN = COS (TH)
   */
   CALL FUNCTION (2,.TH);
   TH  = T;
   TH1 = T1;
   TH2 = T2;

   /*
   TH = SIN (TH) = COS (PI/2 - TH)
   */

   /*
   ANGLE IN QUADRANT 1 OR 3
   */
   IF ((I = 1) OR (I = 3)) THEN DO;
      Z  = (S    := THN);
      Z1 = (S(1) := THN1);
      Z2 = (S(2) := THN2);
      T  = (Y    := TH);
      T1 = (Y(1) := TH1);
      T2 = (Y(2) := TH2);
      IF (I = 3) THEN DO;
         Z2 = (S(2) := Z2 XOR 080H);
         T2 = (Y(2) := T2 XOR 080H);
         RETURN;
         END;
      RETURN;
      END;
```

```
   /*
   ANGLE IN QUADRANT 2 OR 4
   */
   Z  = (S    := TH);
   Z1 = (S(1) := TH1);
   Z2 = (S(2) := TH2 XOR 080H);
   T  = (Y    := THN);
   T1 = (Y(1) := THN1);
   T2 = (Y(2) := THN2);
   IF (I = 4) THEN DO;
      T2 = (S(2) := T2 XOR 080H);
      Z2 = (Y(2) := Z2 XOR 080H);
      END;
   RETURN;
END COSSIN;


ATAN: PROCEDURE (XA,AT);
DECLARE XA ADDRESS, X BASED XA BYTE,
        AT ADDRESS, TV BASED AT BYTE;

   CALL FUNCTION(0,.X);
   TV(0) = T;
   TV(1) = T1;
   TV(2) = T2;
   RETURN;
END ATAN;


REAL: PROCEDURE (AD,FL);
DECLARE AD ADDRESS,XI BASED AD BYTE, Z (40) BYTE,FL BYTE;

   BIDEC: PROCEDURE (AY,FL);
   DECLARE AY ADDRESS, Y BASED AY BYTE,
           (FL,FLAG,I,L,K,SHIFT,R,LASTI,J) BYTE,
           X (16) BYTE, (XX,YY) ADDRESS;

    /*
    BIDEC CONVERTS THE BINARY INTERNAL REAL NUMBER TO A
    DECIMAL NUMBER, WHICH CAN BE PRINTED BY THE MONITOR
    */

    /*
    FL IS A FLAG WHICH INDICATES THE NUMBER TO BE
    CONVERTED IS     FL = 0  REAL
                     FL = 1  ADDRESS VARIABLE
                     FL = 2  BYTE VARIABLE
    */

      DO I=0 TO 15;
         X(I)=0;
         END;

      DO I=0 TO 39;
         Z(I)=0;
         END;

      IF Y=0 THEN
         RETURN;

      YY=SHL(DOUBLE(Y),8) OR DOUBLE(Y(1));
      IF (Y(2) AND 80H)<>0 THEN DO;

         FLAG=1;
         K=Y(2) AND 07FH;
         END;
      ELSE DO;
         FLAG=0;
         K=Y(2);
         END;
      L=K-64;
```

```
      IF (K >= 64) THEN DO;
         I=SHR(L,3);
         R=L-SHL(I,3);
         R=8-R;
         I=I+8;
         END;
      ELSE DO;
         L=-L;
         I=SHR(L,3);
         R=L-SHL(I,3);
         I=7-I;
         END;

      IF (I >= 1) THEN DO;
         X(I)=HIGH(YY);
         X(I-1)=LOW(YY);
         YY=SHL(DOUBLE(X(I-1)),8);
         YY=SHR(YY,R);
         IF (I >= 2) THEN
            X(I-2) = LOW(YY);
         YY=SHL(DOUBLE(X(I)),8) OR DOUBLE(X(I-1));
         YY=SHR(YY,R);
         X(I)=HIGH(YY);
         X(I-1)=LOW(YY);
         END;
      ELSE
         X(I)=SHR(X(I),R);

      IF (I >= 8) THEN DO;
         LASTI=I;
         R=0;
         K=20;
RECYCLE: XX=SHL(DOUBLE(R),8) OR X(I);
         YY=XX/10;
         X(I)=LOW(YY);
         R=XX MOD 10;
         IF (I = 8) THEN DO;
            Z(K)=R+'0';
            K=K+1;
            IF (X(8) = 0) THEN
               GO TO FRACT;
            IF (X(LASTI) = 0) THEN
               LASTI = LASTI - 1;
            R=0;
            I=LASTI;
            GO TO RECYCLE;
            END;
         I = I - 1;
         GO TO RECYCLE;
         END;

DECLARE LEASTI BYTE;

FRACT: IF FL=0 THEN DO;
          LEASTI=I-2;
          IF LEASTI >= 8 THEN
             RETURN;
          DO I=0 TO 19;
             L=0;
             DO K=LEASTI TO 7;
                YY=DOUBLE(X(K))*10+DOUBLE(L);
                X(K)=LOW(YY);
                L=HIGH(YY);
                END;
             Z(19-I)=L+'0';
             END;
          END;
       IF FLAG=1 THEN
          Z(39)=0FFH;
       RETURN;
   END EIDEC;
```

```
   PRINTNUM: PROCEDURE;
   DECLARE (I,K) BYTE;
   DECLARE (DIGIT, ENTRY) LABEL;

    /*
    PRINTS A REAL NUMBER IN THE FORM (-)XXXXX.XXXXXXX
    WITH LEADING ZEROES SUPPRESSED
    */

      DO I=1 TO 5;
         IF (I = 5) AND (Z(20) = 0) THEN DO;
            IF (Z(39) = OFFH) THEN
               CALL PRINTCHAR('-');
            ELSE
               CALL PRINTCHAR(' ');
            GO TO ENTRY;
            END;

         IF Z(25-I) <> 0 THEN DO;
            IF (Z(39) = OFFH) THEN
               CALL PRINTCHAR('-');
            ELSE
               CALL PRINTCHAR(' ');
            GO TO DIGIT;
            END;

         CALL PRINTCHAR(' ');
         END;
      GO TO ENTRY;

DIGIT:
      DO K=I TO 5;
         CALL PRINTCHAR(Z(25-K));
         END;

ENTRY:
      CALL PRINTCHAR('.');
      DO K = 1 TO 7;
         CALL PRINTCHAR(Z(20-K));
      END;
      RETURN;
   END PRINTNUM;

   CALL BIDEC(.XI,FL);
   CALL PRINTNUM;
   RETURN;
END REAL;
```

```
 /*
 INITIALIZATION OF OMEGA STATION DIRECTION COSINE VALUES
 WHICH REPRESENTS A UNIT VECTOR FROM EARTH'S CENTER TO
 THE STATION IN GEOCENTRIC COORDINATES
 */

DECLARE              /* STATION A */
SA1 (3) BYTE INITIAL (0EAH, 05FH, 040H),   /*  0.91551829 */
SA2 (3) BYTE INITIAL (0C8H, 08FH, 03FH),   /*  0.39172334 */
SA3 (3) BYTE INITIAL (0B8H, 077H, 03DH),   /*  0.09153733 */

                     /* STATION B */
SB1 (3) BYTE INITIAL (0FBH, 072H, 040H),   /*  0.98221041 */
SB2 (3) BYTE INITIAL (0FBH, 02EH, 040H),   /*  0.98117517 */
SB3 (3) BYTE INITIAL (0A9H, 0DDH, 0BEH),   /* -0.16588382 */

                     /* STATION C */
SC1 (3) BYTE INITIAL (0B9H, 0C5H, 03FH),   /*  0.36283281 */
SC2 (3) BYTE INITIAL (0DCH, 0EBH, 0C0H),   /* -0.86296869 */
SC3 (3) BYTE INITIAL (0B4H, 007H, 0BFH),   /* -0.35162108 */

                     /* STATION D */
SD1 (3) BYTE INITIAL (0B8H, 0B0H, 040H),   /*  0.72144222 */
SD2 (3) BYTE INITIAL (0CDH, 099H, 0BDH),   /* -0.10039077 */
SD3 (3) BYTE INITIAL (0AFH, 066H, 0C0H),   /* -0.68515898 */

                     /* STATION E */
SE1 (3) BYTE INITIAL (0B6H, 032H, 08FH),   /* -0.35585299 */
SE2 (3) BYTE INITIAL (088H, 03AH, 040H),   /*  0.53214875 */
SE3 (3) BYTE INITIAL (0C4H, 0ABH, 040H),   /*  0.76823587 */

                     /* STATION F */
SF1 (3) BYTE INITIAL (0AEH, 023H, 0C0H),   /* -0.68022941 */
SF2 (3) BYTE INITIAL (09DH, 078H, 03FH),   /*  0.30756230 */
SF3 (3) BYTE INITIAL (0AAH, 054H, 0C0H),   /* -0.66535207 */

                     /* STATION G */
SG1 (3) BYTE INITIAL (0BCH, 0ECH, 03EH),   /*  0.18449565 */
SG2 (3) BYTE INITIAL (0EFH, 009H, 03FH),   /*  0.46687116 */
SG3 (3) BYTE INITIAL (0DDH, 067H, 0C0H),   /* -0.86486570 */

                     /* STATION H */
SH1 (3) BYTE INITIAL (090H, 0C2H, 040H),   /*  0.56547261 */
SH2 (3) BYTE INITIAL (086H, 02BH, 0C0H),   /* -0.52409936 */
SH3 (3) BYTE INITIAL (0A3H, 0D7H, 040H);   /*  0.63683639 */


 /*
 VELOCITY AND NAVIGATION PROCESSING DEFINITIONS
 */

DECLARE
        /* NAVIGATION CYCLE - 1.0 SEC */
DTNAV (3) BYTE INITIAL (080H, 000H, 041H),

        /* ANGULAR ROTATIONS ABOUT THE SYSTEM AXIS */
DELT2 (3) BYTE,
DELT3 (3) BYTE,

        /* EARTHS ELLIPTICITY CONSTANT - 0.0033529 */
EEC   (3) BYTE INITIAL (0DBH, 0BCH, 038H),

        /* OMEGA AIRCRAFT LATITUDE AND LONGITUDE   */
OLATAC(3) BYTE,
OLONAC(3) BYTE,
```

```
        /*
        INITIAL LOCATION MONTEREY, CALIFORNIA
        LATITUDE 36DEG 35 MIN N - 0.638499618 RADIANS
        LONGITUDE 121DEG 51 MIN W - 2.126683694 RADIANS
        NOTE: INSERT N LAT. POSITIVE
              INSERT S LAT. NEGATIVE
              INSERT E LONG. POSITIVE
              INSERT W LONG. 360DEG - INITIAL LONG.
        */
CLATIN(3) BYTE INITIAL (0A3H, 074H, 040H),

        /*
        INSERT 360DEG - 121DEG 51MIN W
        INSERT 238DEG 09 MIN - 4.156501614 RADIANS
        */
CLONIN(3) BYTE INITIAL (085H, 002H, 043H),

        /* EARTHS MEAN EQUATORIAL RADIUS - 20925741.47 FT */
EMER  (3) BYTE INITIAL (09FH, 0A6H, 059H),

        /* WIND DIRECTION */
WD    (3) BYTE,

        /* AIRCRAFT HEADING */
ACHDG (3) BYTE,

        /* SYSTEM HEADING ANGLE */
SHDGA (3) BYTE,

        /* OLD MODE, NEW MODE, NAVIGATION MODE */
(OLDM, NEWM, NAVMODE) BYTE,

        /* DVC3 = (VC3 * DTNAV) / EMER */
DVC3 (3) BYTE,

        /* DVC2 = (VC2 * DTNAV) /EMER */
DVC2 (3) BYTE,

        /* ELEMENTS X5 AND X6 OF STATE VECTOR */
SIGVC2 (3) BYTE INITIAL (000H, 000H, 04CH),
SIGVC3 (3) BYTE INITIAL (000H, 000H, 040H),

        /* CONVERSION FACTOR RADIANS TO DEGREES          */
RADTODEG (3) BYTE INITIAL (08EH, 0FAH, 03BH),

        /* TEMP. STORAGE AFTER CONVERSION RAD TO DEG */
RESULT   (3) BYTE,

VAN   (3) BYTE,          /* AIRDATA VELOCITY COMPONENT NORTH*/
VAE   (3) BYTE,          /* AIRDATA VELOCITY COMPONENT EAST */
VDN   (3) BYTE,          /* DOPPLER VELOCITY COMPONENT NORTH*/
VDE   (3) BYTE,          /* DOPPLER VELOCITY COMPONENT EAST */
VIN   (3) BYTE,          /*INERTIAL VELOCITY COMPONENT NORTH*/
VIE   (3) BYTE,          /*INERTIAL VELOCITY COMPONENT EAST */
VC2   (3) BYTE,          /* CORRECTED VELOCITY ALONG R2 AXIS*/
VC3   (3) BYTE,          /* CORRECTED VELOCITY ALONG R3 AXIS*/
VDA   (3) BYTE,          /* DOPPLER VELOCITY ALONG HEADING  */
VDC   (3) BYTE,          /* DOPPLER VELOCITY ACROSS HEADING */
VTAS  (3) BYTE,          /* TRUE AIR SPEED VELOCITY         */
VW    (3) BYTE,          /* WIND VELOCITY                   */

        /* ITEMS OF 3 BY 3 POSITION MATRIX */
R11   (3) BYTE,
R12   (3) BYTE,
R13   (3) BYTE,
R21   (3) BYTE,
R22   (3) BYTE,
R23   (3) BYTE,
R31   (3) BYTE,
R32   (3) BYTE,
R33   (3) BYTE;
```

```
INITIALIZE$R: PROCEDURE;
 /*
 INITIALIZATION OF POSITION MATRIX R
 */

    /*
    R11 = SIN(LAT), R31 = COS(LAT)
    */
    CALL COSSIN (.CLATIN, .R31, .R11);

    /*
    R22 = SIN(LONG), R23 = COS(LONG)
    */
    CALL COSSIN (.CLONIN, .R23, .R22);

    /*
    R12 = COS(LAT) * COS(LONG)
    */
    CALL MULT (.R31, .R23, .R12);

    /*
    R13 = COS(LAT) * SIN(LONG)
    */
    CALL MULT (.R31, .R22, .R13);

    /*
    R21 INITIALIZE TO ZERO
    */
    R21 (0) = 00H;
    R21 (1) = 00H;
    R21 (2) = 00H;

    /*
    CHANGE THE SIGN OF R11
    */
    R11 (2) = R11 (2) XOR 080H;

    /*
    R33 = -SIN(LAT) * SIN(LONG)
    */
    CALL MULT (.R11, .R22, .R33);

    /*
    R32 = -SIN(LAT) * COS(LONG)
    */
    CALL MULT (.R11, .R23, .R32);

    /*
    CHANGE THE SIGN OF R11 AND R22
    */
    R11 (2) = R11(2) XOR 080H;
    R22 (2) = R22(2) XOR 080H;
    RETURN;
END INITIALIZE$R;

NEWPOS: PROCEDURE;

 /*
 UPDATE THE POSITION MATRIX R AND CALCULATE NEW
 LATITUDE, LONGITUDE, AND SYSTEM HEADING ANGLE
 */

DECLARE (A, A1, A2, A3, B, B1, B2, B3, C1, C2, C3) (3) BYTE,
        (SUM1, SUM2, SUM3, D2SQR, D3SQR) (3) BYTE;

   CALL MULT (.DELT2, .R31, .A);
   CALL MULT (.DELT3, .R21, .B);
   CALL SUB (.B, .A, .C1);
```

```
   CALL MULT (.DELT2, .R32, .A);
   CALL MULT (.DELT3, .R22, .B);
   CALL SUB (.B, .A, .C2);

   CALL MULT (.DELT2, .R33, .A);
   CALL MULT (.DELT3, .R23, .B);
   CALL SUB (.B, .A, .C3);

   CALL ADD (.C1, .R11, .SUM1);
   CALL ADD (.C2, .R12, .SUM2);
   CALL ADD (.C3, .R13, .SUM3);

   /*
   SUBTRACTING ONE FROM THE EXPONENT VALUE WILL
   CAUSE THE MANTISSA TO APPEAR TO BE DIVIDED BY 2
   */
   C1 (2) = C1 (2) - 001H;
   C2 (2) = C2 (2) - 001H;
   C3 (2) = C3 (2) - 001H;

   CALL ADD (.C1, .R11, .A1);
   CALL ADD (.C2, .R12, .A2);
   CALL ADD (.C3, .R13, .A3);

   CALL MULT (.DELT2, .DELT2, .D2SQR);
   CALL MULT (.DELT3, .DELT3, .D3SQR);
   CALL ADD (.D2SQR, .D3SQR, .B);

   B (2) = B (2) - 001H;
   CALL MULT (.B, .R11, .B1);
   CALL MULT (.B, .R12, .B2);
   CALL MULT (.B, .R13, .B3);

   CALL SUB (.SUM1, .B1, .R11);
   CALL SUB (.SUM2, .B2, .R12);
   CALL SUB (.SUM3, .B3, .R13);

   CALL MULT (.A1, .DELT3, .A);
   CALL MULT (.A1, .DELT2, .B);
   CALL SUB (.R21, .A, .R21);
   CALL ADD (.R31, .B, .R31);

   CALL MULT (.A2, .DELT3, .A);
   CALL MULT (.A2, .DELT2, .B);
   CALL SUB (.R22, .A , .R22);
   CALL ADD (.R32, .B, .R32);

   CALL MULT (.A3, .DELT3, .A);
   CALL MULT (.A3, .DELT2, .B);
   CALL SUB (.R23, .A, .R23);
   CALL ADD (.R33, .B, .R33);

   CALL DIV (.R21, .R31, .A);
   CALL DIV (.R13, .R12, .B);

   CALL ATAN (.A, .SHDGA);
   CALL ATAN (.B, .OLONAC);

   CALL CCSSIN (.SHDGA, .B, .A);

   CALL MULT (.B, .R11, .A);
   CALL DIV (.A, .R31, .B);
   CALL ATAN (.B, .OLATAC);
   RETURN;
END NEWPCS;

VELPROC: PROCEDURE;
 /*
 COMPUTE NORTH, SOUTH, EAST, AND WEST VELOCITIES
 AND COMPUTE VELOCITIES ALONG THE SYSTEM AXIS
 */
```

```
DECLARE (A, A1, A2, A3) (3) BYTE,
        (B, B1, B2, B3) (3) BYTE,
        (C, C1, C3, D, D1) (3) BYTE,
        (DELVC2, DELVC3) (3) BYTE,
        (VELE, VELN) (3) BYTE;

   /*
   B = COS(AIRCRAFT HEADING), A = SIN(AIRCRAFT HEADING)
   */
   CALL CCSSIN (.ACHDG, .B, .A);

   /*
   D = COS(WIND DIRECTION), C = SIN(WIND DIRECTION)
   */
   CALL CCSSIN (.WD, .D, .C);

   /*
   A1 = SIN(ACHDG) * VDC
   */
   CALL MULT (.A, .VDC, .A1);

   /*
   B1 = COS(ACHDG) * VDA
   */
   CALL MULT (.B, .VDA, .B1);

   /*
   A2 = SIN(ACHDG) * VDA
   */
   CALL MULT (.A, .VDA, .A2);

   /*
   B2 = COS(ACHDG) * VDC
   */
   CALL MULT (.B, .VDC, .B2);

   /*
   VDN = COS(ACHDG) * VDA - SIN(ACHDG) * VDC
   */
   CALL SUB (.B1, .A1, .VDN);

   /*
   VDE = COS(ACHDG) * VDC + SIN(ACHDG) * VDA
   */
   CALL ADD (.B2, .A2, .VDE);

   /*
   B3 = COS(ACHDG) * VTAS
   */
   CALL MULT (.B, .VTAS, .B3);

   /*
   A3 = SIN(ACHDG) * VTAS
   */
   CALL MULT (.A, .VTAS, .A3);

   /*
   C1 = COS(WD) * VW
   */
   CALL MULT (.D, .VW, .D1);

   /*
   C1 = SIN(WD) * VW
   */
   CALL MULT (.C, .VW, .C1);

   /*
   VAN = COS(ACHDG) * VTAS + COS(WD) * VW
   */
   CALL ADD (.B3, .D1, .VAN);
```

```
/*
VAE = SIN(ACHDG) * VTAS + SIN(WD) * VW
*/
CALL ADD (.A3, .C1, .VAE);

/*
B = COS(SHDGA), A = SIN(SHDGA)
*/
CALL CCSSIN (.SHDGA, .B, .A);

   DO CASE NAVMODE;
   /* CASE 0 - INERTIAL COMPONENTS */
      DO;
      VELN(0) = VIN(0);
      VELN(1) = VIN(1);
      VELN(2) = VIN(2);
      VELE(0) = VIE(0);
      VELE(1) = VIE(1);
      VELE(2) = VIE(2);
      END;

   /* CASE 1 - DOPPLER COMPONENTS */
      DO;
      VELN(0) = VDN(0);
      VELN(1) = VDN(1);
      VELN(2) = VDN(2);
      VELE(0) = VDE(0);
      VELE(1) = VDE(1);
      VELE(2) = VDE(2);
      END;

   /* CASE 2 - AIR DATA COMPONENTS */
      DO;
      VELN(0) = VAN(0);
      VELN(1) = VAN(1);
      VELN(2) = VAN(2);
      VELE(0) = VAE(0);
      VELE(1) = VAE(1);
      VELE(2) = VAE(2);
      END;
   END;

   /*
   B1 = VELE * COS(SHDGA)
   */
   CALL MULT (.VELE, .B, .B1);

   /*
   A1 = VELN * SIN(SHDGA)
   */
   CALL MULT (.VELN, .A, .A1);

   /*
   B2 = VELN * COS(SHDGA)
   */
   CALL MULT (.VELN, .B, .B2);

   /*
   A2 = VELE * SIN(SHDGA)
   */
   CALL MULT (.VELE, .A, .A2);

   /*
   A3 = VELE * COS(SHDGA) + VELN * SIN(SHDGA)
   */
   CALL ADD (.B1, .A1, .A3);

   /*
   B3 = VELN * COS(SHDGA) - VELE * SIN(SHDGA)
   */
   CALL SUB (.B2, .A2, .B3);
```

```
   /*
   CHECK IF UPGRADE OR DOWNGRADE OF NAV. MODE
   */

   IF OLDM < NEWM THEN
      DO;
      OLDM = NEWM;

      /*
      DELTA VC2 = VC2 - (VELE*COS(SHDGA) + VELN*SIN(SHDGA))
      */
      CALL SUB (.VC2, .A3, .DELVC2);

      /*
      DELTA VC3 = VC3 - (VELN*COS(SHDGA) - VELE*SIN(SHDGA))
      */
      CALL SUB (.VC3, .B3, .DELVC3);
      END;
   ELSE
      DO;
      IF OLDM > NEWM THEN
         DO;
         /*
         SET DELTA VC2 AND DELTA VC3 TO ZERO
         */
         DELVC2 (0), DELVC3 (0) = 000H;
         DELVC2 (1), DELVC3 (1) = 000H;
         DELVC2 (2), DELVC3 (2) = 040H;
         END;
      OLDM = NEWM;
      CALL SUB (.DELVC2, .SIGVC2, .DELVC2);
      CALL SUB (.DELVC3, .SIGVC3, .DELVC3);
      END;

   /*
   SET SIGMA VC2 AND SIGMA VC3 TO ZERO
   */
   SIGVC2 (0), SIGVC3 (0) = 000H;
   SIGVC2 (1), SIGVC3 (1) = 000H;
   SIGVC2 (2), SIGVC3 (2) = 040H;

   CALL ADD (.A3, .DELVC2, .VC2);
   CALL ADD (.B3, .DELVC3, .VC3);
   RETURN;
END VELPROC;


NAVPROC: PROCEDURE;
 /*
 NAVIGATION PROCESSING FOR POSITION COMPUTATIONS
 */

DECLARE (R11SQR, R21SQR, R31SQR, R2131) (3) BYTE,
        (DIFF1, DIFF2, W, X, Y, Z) (3) BYTE,
        (TEMP1, TEMP2, TEMP3) (3) BYTE;
DECLARE ZERO (3) BYTE INITIAL (000H, 000H, 040H),
        ONE  (3) BYTE INITIAL (080H, 000H, 041H);

   /*
   R11SQR = R11 * R11
   */
   CALL MULT (.R11, .R11, .R11SQR);

   /*
   R21SQR = R21 * R21
   */
   CALL MULT (.R21, .R21, .R21SQR);

   /*
   R31SQR = R31 * R31
   */
   CALL MULT (.R31, .R31, .R31SQR);
```

```
   /*
   R2131 = R21 * R31
   */
   CALL MULT (.R21, .R31, .R2131);

   /*
   ADDING ONE TO THE EXPONENT VALUE WILL CAUSE
   THE MANTISSA TO APPEAR TO BE MULTIPLIED BY 2
   */
   R31SQR(2) = R31SQR (2) + 001H;
   R21SQR(2) = R21SQR (2) + 001H;
   R2131 (2) = R2131 (2) + 001H;

   /*
   DIFF1 = 2 * (R31 * R31) - R11 * R11
   */
   CALL SUB (.R31SQR, .R11SQR, .DIFF1);

   /*
   DIFF2 = 2 * (R21 * R21) - R11 * R11
   */
   CALL SUB (.R21SQR, .R11SQR, .DIFF2);

   /*
   TEMP1 = DIFF1 * EEC
   TEMP2 = DIFF2 * EEC
   TEMP3 = R2131 * EEC
   */
   CALL MULT (.DIFF1, .EEC, .TEMP1);
   CALL MULT (.DIFF2, .EEC, .TEMP2);
   CALL MULT (.R2131, .EEC, .TEMP3);

   /*
   W = 1 + EEC*(2*R31*R31 - R11*R11)
   X = 1 + EEC*(2*R21*R21 - R11*R11)
   */
   CALL ADD (.TEMP1, .ONE, .W);
   CALL ADD (.TEMP2, .ONE, .X);

   /*
   TEMP1 = DVC3 * (1+EEC*(2*R31*R31 - R11*R11))
   TEMP2 = DVC2 * (1+EEC*(2*R21*R21 - R11*R11))
   Y     = DVC3 * (EEC*(2*R21*R31))
   Z     = DVC2 * (EEC*(2*R21*R31))
   */
   CALL MULT (.W, .DVC3, .TEMP1);
   CALL MULT (.X, .DVC2, .TEMP2);
   CALL MULT (.TEMP3, .DVC3, .Y);
   CALL MULT (.TEMP3, .DVC2, .Z);

   /*
   CHANGE THE SIGN OF TEMP1
   TEMP1 = -DVC3*(1+EEC*(2*R31*R31 - R11*R11))
   */
   TEMP1 (2) = TEMP1 (2) XOR 080H;

   /*
   DELT2 = -DVC3*(1+EEC*(2*R31*R31-R11*R11))
           - DVC2*(EEC*(2*R21*R31))
   */
   CALL SUB (.TEMP1, .Z, .DELT2);

   /*
   DELT3 =  DVC2*(1+EEC*(2*R21*R21-R11*R11)) +
            DVC3*(EEC*(2*R21*R31))
   */
   CALL ADD (.TEMP2, .Y, .DELT3);
   RETURN;
END NAVFRCC;
```

```
OMEGA: PROCEDURE;
DECLARE (I,J,K) BYTE,
        (TMPONE, TMPTWO) (3) BYTE,
        (SUMDELT2, SUMDELT3) (3) BYTE;
DECLARE PI (3) BYTE INITIAL (0C9H, 00FH, 042H);

   CALL INITIALIZE$R;
   DO K = 1 TO 60;
      DO J = 1 TO 4;
         SUMDELT2(0), SUMDELT3(0) = 0C0H;
         SUMDELT2(1), SUMDELT3(1) = 000H;
         SUMDELT2(2), SUMDELT3(2) = 040H;
         DO I = 1 TO 15;
            CALL VELPROC;

            /*
            DVC3 = (VC3 * DTNAV)/EMER
            DVC2 = (VC2 * DTNAV)/EMER
            */
            CALL MULT (.VC3, .DTNAV, .TMPONE);
            CALL DIV (.TMPONE, .EMER, .DVC3);
            CALL MULT (.VC2, .DTNAV, .TMPTWO);
            CALL DIV (.TMPTWO, .EMER, .DVC2);
            CALL NAVPROC;
            CALL ADD (.SUMDELT2, .DELT2, .SUMDELT2);
            CALL ADD (.SUMDELT3, .DELT3, .SUMDELT3);
            END;

         DELT2(0) = SUMDELT2(0);
         DELT2(1) = SUMDELT2(1);
         DELT2(2) = SUMDELT2(2);
         DELT3(0) = SUMDELT3(0);
         DELT3(1) = SUMDELT3(1);
         DELT3(2) = SUMDELT3(2);
         CALL NEWPOS;
         CALL PRINT (.'     OMEGA LONGITUDE-RADIANS$');
         CALL REAL (.OLONAC, 0);
         CALL DIV (.OLONAC, .RADTODEG, .RESULT);
         CALL PRINT (.'     OMEGA LONGITUDE-DEGREES$');
         CALL REAL (.RESULT, 0);
         END;

      END;
   RETURN;
END OMEGA;


 /* INITIAL CONDITIONS - NO WIND SOLUTION FLYING DUE EAST
    AT 300 KNOTS FROM MONTEREY, CALIF. AIRPORT */

OLDM = 0;
NEWM = 2;
NAVMODE = 2;

 /* WIND DIRECTION 000 DEG = 0.000000RAD */
WD(0) = 000H;
WD(1) = 0C0H;
WD(2) = 000H;

 /* AIRCRAFT HEADING 090 DEG = 1.570788 RAD */
ACHDG(0) = 0C9H;
ACHDG(1) = 00FH;
ACHDG(2) = 041H;

 /* SYSTEM HEADING ANGLE 000 DEG = 0.000000 RAD */
SHDGA(0) = 000H;
SHDGA(1) = 000H;
SHDGA(2) = 000H;

VAN(0), VAE(0), VDN(0), VDE(0), VIN(0), VIE(0) = 000H;
VAN(1), VAE(1), VDN(1), VDE(1), VIN(1), VIE(1) = 000H;
VAN(2), VAE(2), VDN(2), VDE(2), VIN(2), VIE(2) = 000H;
```

```
VC3(C), VCA(C), VCC(0), VW(0) = 000H;
VC3(1), VCA(1), VDC(1), VW(1) = 000H;
VC3(2), VCA(2), VDC(2), VW(2) = 000H;

 /*
 TRUE AIR SPEED = 300 KNOTS
 300 KNOTS = 506.3429575 FT/SEC
 */
VTAS(C) = 0FDH;
VTAS(1) = 02BH;
VTAS(2) = 049H;

 /*
 SET CORRECTED VELOCITY = 300 KNOTS
 300 KNOTS = 506.3429575 FT/SEC
 */
VC2(C) = 0FDH;
VC2(1) = 02BH;
VC2(2) = 049H;

 /* MAIN PROGRAM */
CALL OMEGA;
HALT;

EOF
```

LIST OF REFERENCES

1. Culver, C. O. and Danklefs, R. W., "The Use of Microprocessors in Navigation Systems," Navigation, v. 23, no. 3, p. 245-248, Fall 1976.

2. Gaon, B. N., "Hand Held Calculator Technology Applied to an Advanced Omega Receiver," Navigation, v. 22, no. 4, p. 302-308, Winter 1975-1976.

3. Delorme, J. F. and Tuppen, A. R., "Low-Cost Navigation Processing for Loran-C and Omega," Navigation, v. 22, no. 2, p. 112-127, Summer 1975.

4. Naval Air Development Center, P-3C Update System Functional Description for Omega, FD-107(I.2), p. 1-217, September 1976.

5. Smith, E. J., "AN/ARN-99 Method of Removing Omega Lane Ambiguity," In Proceedings of the First Omega Symposium, p. 75-82, November 1971.

6. Commander, Air Test and Evaluation Squadron ONE, OPTEVFOR Tactics Guide (OTG 76-4), OMEGA Navigation in the P-3C UPDATE Weapons System, p. 3-4, 10 May 1976.

7. Rey, J. A. and Sakran, F. C., "The AN/ARN-99(V) Airborne Omega Navigation Set: Capabilities and Status," In Proceedings of the First Omega Symposium, p. 89-98, November 1971.

8. Hawkes, K. H. and Birnbaum, D., "Integration and Flight Test of an Omega Receiver with the P-3C Aircraft Navigation System," In Proceedings of the First Omega Symposium, p. 190-201, November 1971.

9. Brown, R. G. and Hagerman, L. L., "An Optimum Inertial/Doppler Satellite Navigation System," Navigation, v. 16, no. 3, p. 261, Fall 1969.

10. Brown, R. G., "Integrated Navigation Systems and Kalman Filtering: A Perspective," Navigation, v. 19, no. 4, p. 355, Winter 1972-1973.

11. Northrop Corp., Hawthrone, Calif. Electronics Div., Omega Navigation Set AN/ARN-99 (XN-2) (U), Microfiche AD-775598, December 1973.

12. Weinman, R. W., "Real Time Propagation Prediction for an Airborne Omega System," In Proceedings of the First Omega Symposium, p. 83-88, November 1971.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor Uno R. Kodres, Code 52Kr Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Captain James J. Frydrychowicz, USMC 9104 West 28th Street Brookfield, Illinois 60513	1